Θεόδωρος Σαμαράς
Τμήμα Φυσικής
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

ΕΙΣΑΓΩΓΗ ΣΤΗ ΧΡΗΣΗ ΤΟΥ MATLAB®

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Θεσσαλονίκη 2004

ΠΕΡΙΕΧΟΜΕΝΑ

1 Απλοί υπολογισμοί

1.1 Ξεκινώντας

Μετά από την εκκίνηση του MATLAB εμφανίζεται η γραμμή εντολών:

```
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, type tour or visit www.mathworks.com.

»
```

Πληκτρολογώντας `helpwin`, μπορούμε να εμφανίσουμε τον κατάλογο θεμάτων για βοήθεια. Πληκτρολογώντας `demo`, ξεκινά μια εκτεταμένη λειτουργία επίδειξης που παρουσιάζει μερικές αξιοσημείωτες δυνατότητες του προγράμματος. Η έξοδος από το MATLAB γίνεται, πληκτρολογώντας `exit` ή `quit` στη γραμμή εντολών.

1.2 Αριθμητικές πράξεις

Οι τελεστές του MATLAB για τις τέσσερις στοιχειώδεις αριθμητικές πράξεις είναι + για την πρόσθεση, - για την αφαίρεση, * για τον πολλαπλασιασμό και / (καθώς και \) για τη διαίρεση. Μερικά παραδείγματα χρήσης αυτών των τελεστών παρουσίάζονται παρακάτω:

```
» 1+1

ans =

     2

» 1-1

ans =

     0

» 2*3

ans =

     6

» 3/4

ans =

    0.7500
```

Μπορείτε να γράψετε συνδυασμούς πράξεων στη γραμμή εντολών:

```
» 1+2*3
ans =
     7
```

Ο τελεστής ύψωσης σε δύναμη είναι ο ^ μπορεί να χρησιμοποιηθεί για οποιαδήποτε τιμή του εκθέτη:

```
» 3^2
ans =
     9
» 10^(1/3)
ans =
    2.1544
```

Το MATLAB χρησιμοποιεί δύο τελεστές για τη διαίρεση, έναν για διαίρεση προς τα δεξιά / και έναν για διαίρεση προς τα αριστερά \, όπως φαίνεται παρακάτω:

```
» 10/2
ans =
     5
» 2\10
ans =
     5
```

Όταν οι τελεστέοι της διαίρεσης είναι πραγματικοί αριθμοί, τότε το αποτέλεσμα της πράξης είναι ανεξάρτητο του τελεστή που χρησιμοποιούμε. Σε επόμενη ενότητα όμως θα δούμε ότι το MATLAB επεκτείνει το συνήθη ορισμό της διαίρεσης και για διανύσματα και πίνακες. Στις τελευταίες αυτές περιπτώσεις οι διαιρέσεις προς τα δεξιά και τα αριστερά δεν είναι πλέον αντιμεταθέσιμες.

Αν οι αριθμητικές εκφράσεις που θέλουμε να υπολογίζουμε είναι πολύπλοκες τότε από το MATLAB ακολουθείται μια προτεραιότητα στην εκτέλεση των πράξεων από τα αριστερά προς

τα δεξιά. Προηγείται ο τελεστής της ύψωσης σε δύναμη, ακολουθούν σε προτεραιότητα οι τελεστές πολλαπλασιασμού και διαίρεσης

```
» 2^3*4
ans =
    32
```

και έπονται οι τελεστές πρόσθεσης και αφαίρεσης

```
» 2^3+2*12
ans =
    32
```

Αν θέλουμε να υπολογίσουμε πιο περίπλοκες εκφράσεις, αλλάζοντας την προτεραιότητα των πράξεων ή δεν είμαστε σίγουροι για τη σειρά με την οποία αυτές θα εκτελεστούν, τότε χρησιμοποιούμε παρενθέσεις με το συνήθη τρόπο:

```
» 2^((3+2)*2)
ans =
        1024
```

1.3 'Εκδοση τελευταίας γραμμής

Αν υποθέσουμε ότι, πληκτρολογώντας την παραπάνω έκφραση

```
2^((3+2)*2)
```

για να εκτελέσουμε τον αντίστοιχο υπολογισμό έχουμε κάνει λάθος, και αντί για το ψηφίο 2 στο τέλος έπρεπε να πληκτρολογήσουμε το 4, μπορούμε να διορθώσουμε την τελευταία γραμμή, χρησιμοποιώντας το πλήκτρο για επάνω βέλος ↑ στο πληκτρολόγιο. Με τον τρόπο αυτόν εμφανίζεται στη γραμμή εντολών η τελευταία γραμμή που πληκτρολογήσαμε. Με τη χρήση του αριστερού βέλους ← και του δεξιού βέλους → μπορούμε να κινήσουμε το δρομέα (cursor) πάνω στη γραμμή και να σβήσουμε το ψηφίο ή σύμβολο που δεν θέλουμε αριστερά του δρομέα με το πλήκτρο Backspace ή δεξιά του δρομέα με το πλήκτρο Delete. Μετά εισάγουμε κανονικά το σωστό ψηφίο ή σύμβολο στη θέση του δρομέα και πατάμε το πλήκτρο Enter. Μια άλλη πολύ χρήσιμη λειτουργία είναι η δυνατότητα επαναφοράς γραμμών εντολής,

των οποίων γνωρίζουμε την αρχή. Αυτή η επαναφορά υλοποιείται, πληκτροκλογώντας τα πρώτα ψηφία ή σύμβολα της γραμμής που γνωρίζουμε και χρησιμοποιώντας το πλήκτρο με το επάνω βέλος και πάλι, ώσπου να βρούμε τη γραμμή που ψάχνουμε.

1.4 Μερικές ενσωματωμένες συναρτήσεις

Σε έναν υπολογιστή χειρός (τσέπης) η εύρεση της τιμής μιας συνάρτησης γίνεται πληκτρολογώντας πρώτα το όρισμα της συνάρτησης και μετά το πλήκτρο με τη συνάρτηση. Για παράδειγμα, για να βρούμε την τετραγωνική ρίζα του αριθμού 9 σε έναν υπολογιστή χειρός (τσέπης), πρέπει πρώτα να πιέσουμε στο πλήκτρο με τον αριθμό 9 και μετά στο πλήκτρο με το σύμβολο $\sqrt{}$. Αντίθετα, στο MATLAB η συνάρτηση πρέπει να γραφεί πλήρως, γράφοντας το κάθε γράμμα του ονόματός της και στη συνέχεια να ακολουθήσει η τιμή του ορίσματος μέσα σε παρενθέσεις. 'Ετσι, για να υπολογίσουμε την τετραγωνική ρίζα του αριθμού 9, πρέπει να πληκτρολογήσουμε `sqrt(9)` και να πατήσουμε το Enter:

```
» sqrt(9)

ans =

     3
```

Για να επιβεβαιώσουμε τη λειτουργία της συνάρτησης, πληκτρολογούμε:

```
» sqrt(10)^2

ans =

   10.0000
```

Εκτός από την τετραγωνική ρίζα, το MATLAB έχει μια χρήσιμη συλλογή από συναρτήσεις, μερικές από τις οποίες φαίνονται στον Πίνακα 1.

Πίνακας 1: Μερικές ενσωματωμένες αριθμητικές συναρτήσεις

Ονομασία συνάρτησης	Πράξη	Αποτέλεσμα στην περιοχή
sin(x)	ημίτονο του x	-1 ως 1
cos(x)	συνημίτονο του x	-1 ως 1
tan(x)	εφαπτομένη του x	
asin(x)	τόξο ημιτόνου του x	-π/2 ως +π/2
acos(x)	τόξο συνημιτόνου του x	0 ως π
atan(x)	τόξο εφαπτομένης του x	-π/2 ως +π/2
atan2(y,x)	τόξο εφαπτομένης τεσσάρων τεταρτημορίων του y/x	-π ως +π
exp(x)	e^x	
log(x)	νεπέρειος λογάριθμος του x	
log10(x)	δεκαδικός λογάριθμος του x	
log2(x)	λογάριθμος του x με βάση το 2	

Παρακάτω δίνονται ορισμένα παραδείγματα χρήσης των τριγωνομετρικών συναρτήσεων. Η τιμή του π στο MATLAB γράφεται ως `pi`.

```
» sin(pi)

ans =

  1.2246e-016

» cos(pi/3)

ans =

    0.5000

» tan(pi/4)

ans =

    1.0000
```

Μερικές φορές εμφανίζεται ένας πολύ μικρός αριθμός εκεί που θα αναμενόταν το μηδέν (όπως στο πρώτο από τα παραπάνω παραδείγματα). Αυτό οφείλεται σε σφάλματα λόγω των αριθμητικών προσεγγίσεων που είναι αναπόφευκτες στους υπολογιστές. Παρόλα αυτά, δεν είναι σωστό να θεωρήσουμε ότι κάθε μικρός αριθμός θα έπρεπε να είναι μηδέν, αλλά αυτές οι περιπτώσεις θα πρέπει να αντιμετωπίζονται με προσοχή. Από τα παραπάνω

παραδείγματα φαίνεται επίσης ότι τα ορίσματα των τριγωνομετρικών συναρτήσεων δίνονται πάντα σε ακτίνια. Αν γνωρίζουμε την τιμή της γωνίας σε μοίρες, τότε θα πρέπει να την μετατρέψουμε κατάλληλα. Για παράδειγμα για να βρούμε το ημίτονο των 50° γράφουμε

```
» sin(pi*50/180)
ans =
    0.7660
```

Επίσης μπορούμε να χρησιμοποιήσουμε μια άλλη ενσωματωμένη συνάρτηση του MATLAB για την μετατροπή, η οποία καλείται πληκτρολογώντας `deg2rad(x)`:

```
» sin(deg2rad(50))
ans =
    0.7660
```

Το αποτέλεσμα των αντίστροφων τριγωνομετρικών συναρτήσεων είναι επίσης σε ακτίνια:

```
» asin(0)
ans =
     0
» acos(0)
ans =
    1.5708
» atan(1)
ans =
    0.7854
```

Τα αποτελέσματα των αντίστροφων τριγωνομετρικών συναρτήσεων μπορούν επίσης εύκολα να μετατραπούν σε μοίρες είτε με την αντιστοιχία του π είτε χρησιμοποιώντας τη συνάρτηση `rad2deg(x)`, ως ακολούθως:

```
» acos(0)*180/pi
```

```
ans =

    90

» rad2deg(acos(0))

ans =

    90
```

Δίνονται και ορισμένα παραδείγματα για τις λογαριθμικές συναρτήσεις:

```
» log(exp(2))

ans =

     2

» log10(1000)

ans =

    3.0000

» 10^log10(100)

ans =

   100

» log2(2^10)

ans =

    10
```

1.5 Ορισμός σταθερών και μεταβλητών

Στην περίπτωση που θέλουμε να υπολογίσουμε την τιμή μιας συνάρτησης για ένα όρισμα που με τη σειρά του αποτελεί μια πολύπλοκη αριθμητική έκφραση είναι καλό να χρησιμοποιήσουμε μεταβλητές. Οι μεταβλητές είναι συμβολικά ονόματα που παίρνουν κάποια αριθμητική τιμή και χρησιμοποιούνται στις μαθηματικές εκφράσεις αντί της τιμής αυτής.

Μια μεταβλητή είναι η `ans`, η οποία εμφανίζεται σε όλα τα παραπάνω παραδείγματα πριν από την αριθμητική τιμή του αποτελέσματος. Στη μεταβλητή αυτή σώζεται αυτόματα από το MATLAB η τιμή του αποτελέσματος του τελευταίου υπολογισμού που εκτελέστηκε. Ως

παράδειγμα, ας υπολογίσουμε το ημίτονο της γωνίας 50°, αφού πρώτα τη μετατρέψουμε σε ακτίνια:

```
» 50*pi/180
ans =
    0.8727
» sin(ans)
ans =
    0.7660
```

Παρατηρούμε ότι ο συντελεστής μετατροπής μια γωνίας από μοίρες σε ακτίνια μπορεί να χρειαστεί πολλές φορές. Μπορούμε να τον καταχωρήσουμε σε μια μεταβλητή, έστω `d2r`, και να τον χρησιμοποιήσουμε σε όλες τις περιπτώσεις που θα χρειαστεί. Για παράδειγμα, αν θέλουμε να υπολογίσουμε το ημίτονο των γωνιών 50° και 65°, μπορούμε τις τιμές αυτών των δυο γωνιών να τις αποδώσουμε σε ισάριθμες μεταβλητές τις gwnia1 και gwnia2:

```
» d2r=pi/180
d2r =
    0.0175
» gwnia1=50
gwnia1 =
    50
» gwnia2=65;
```

Σχετικά με τα ονόματα των μεταβλητών, παρατηρούμε ότι αυτά μπορούν να περιλαμβάνουν μόνο λατινικούς χαρακτήρες, τα αριθμητικά ψηφία (0-9) και την κάτω παύλα _ (underscore). Τα ονόματα ξεκινούν πάντα με λατινικό ψηφίο και πρέπει να τα διαλέγουμε προσεκτικά, γιατί ορισμένα από αυτά μπορεί να συμπέσουν με ονόματα ενσωματωμένων συναρτήσεων (π.χ. `tan`), μεταβλητών (π.χ. `ans`) ή σταθερών (π.χ. `pi`) του MATLAB.

Στο τελευταίο από τα παραπάνω παραδείγματα φαίνεται ότι, αν χρησιμοποιήσουμε το ελληνικό ερωτηματικό ; (semi-colon) στο τέλος μια γραμμής, το αποτέλεσμα της εντολής δε φαίνεται (όλες οι παραπάνω εντολές είναι εντολές καταχώρησης τιμής σε μεταβλητή). Συνεχίζοντας το παράδειγμα, μπορούμε να υπολογίσουμε τις τιμές του ημιτόνου ως εξής:

```
» sin(d2r*gwnia1)

ans =

    0.7660

» sin(d2r*gwnia2)

ans =

    0.9063
```

Κάνοντας ακόμα ένα βήμα, μπορούμε να καταχωρήσουμε τις τιμές του ημιτόνου και του συνημιτόνου μιας γωνίας σε δύο μεταβλητές και να επιβεβαιώσουμε μια γνωστή τριγωνομετρική ταυτότητα:

```
» a=sin(d2r*gwnia1);
» b=cos(d2r*gwnia1);
» a^2+b^2

ans =

     1
```

Το τελευταίο στοιχείο που αξίζει να τονιστεί είναι ότι, αν και στα ονόματα των μεταβλητών στο MATLAB μόνο οι πρώτοι 19 χαρακτήρες είναι σημαντικοί, υπάρχει διαφορά ανάμεσα στα πεζά και τα κεφαλαία γράμματα (όπως συμβαίνει για τα ονόματα μεταβλητών και στις γλώσσες προγραμματισμού C και C++):

```
» a=1;
» A=2;
» 2*a

ans =

     2

» 2*A

ans =

     4
```

1.6 Μορφή εμφάνισης αριθμών

Είδαμε παραπάνω ότι η σταθερά π συμβολίζεται στο MATLAB με τη σταθερά `pi`. Αν πληκτρολογήσουμε `pi`, τότε παίρνουμε μια τιμή για τη σταθερά π με τέσσερα δεκαδικά ψηφία:

```
» pi

ans =

    3.1416
```

Στην πραγματικότητα όμως η σταθερά αυτή σώζεται στο MATLAB με μεγαλύτερη ακρίβεια, με την οποία χρησιμοποιείται και στους υπολογισμούς. Μπορούμε να εμφανίσουμε μια ακριβέστερη τιμή για τη σταθερά π, γράφοντας

```
» format long
» pi

ans =

   3.14159265358979
```

Επομένως, είναι δυνατό να αλλάξουμε τη μορφή με την οποία εμφανίζονται οι αριθμητικές μεταβλητές και σταθερές στο MATLAB, χρησιμοπιώντας την εντολή `format`. Μπορούμε να επιστρέψουμε στην προεπιλεγμένη μορφή αριθμών πληκρολογώντας απλά `format`:

```
» format
» pi

ans =

    3.1416
```

Περισσότερες πληροφορίες για τις δυνατότητες της εντολής `format` μπορούμε να πάρουμε, πληκτρολογώντας `help format`.

2 Μονοδιάστατες σειρές και γραφικά

2.1 Εισαγωγή μονοδιάστατων σειρών

Μία διατεταγμένη συλλογή αριθμών μπορεί καταχωρηθεί στο MATLAB σε μια σειρά. Για παράδειγμα η σειρά των πέντε πρώτων περιττών αριθμών μπορεί να εισαχθεί ως

```
» odds=[1 3 5 7 9]

odds =

     1     3     5     7     9
```

Αντί για κενά μεταξύ των αριθμών, μπορούν να χρησιμοποιηθούν κόμματα για την εισαγωγή των στοιχείων της σειράς:

```
» odds=[1,3,5,7,9]

odds =

     1     3     5     7     9
```

Τα στοιχεία μια σειράς προσδιορίζονται από το δείκτη τους, ο οποίος δηλώνει τη θέση τους στη διατεταγμένη συλλογή. Οι δείκτες είναι φυσικοί αριθμού και σε αντίθεση με τις γλώσσες προγραμματισμού C/C++ ξεκινούν πάντα από την τιμή 1. Για παράδειγμα:

```
» odds(1)

ans =

     1

» odds(3)

ans =

     5
```

Μια σειρά μπορεί να εισαχθεί και στοιχείο προς στοιχείο, με χρήση των δεικτών. Για παράδειγμα, μπορούμε να εισάγουμε τους πρώτους δύο άρτιους αριθμούς ως εξής:

```
» evens(1)=0

evens =
```

```
     0
» evens(2)=2
evens =
     0     2
```

Αν θέλουμε, μπορούμε επίσης να παραλείψουμε μια θέση της σειράς και να δώσουμε τιμή σε κάποια επόμενη θέση. Στην περίπτωση αυτή το MATLAB αυτόματα δίνει την τιμή 0 στη θέση που παραλείφθηκε. Για παράδειγμα, στη σειρά `evens` έστω ότι θέλουμε να παραλείψουμε για κάποιο λόγο τον τρίτο άρτιο αριθμό, δηλαδή το 4, και να εισάγουμε τον τέταρτο, δηλαδή το 6:

```
» evens(4)=6
evens =
     0     2     0     6
```

Ένας ακόμα τρόπος για την εισαγωγή των σειρών (ιδιαίτερα όταν αυτές είναι απλές αριθμητικές ακολουθίες) γίνεται με το πρώτο και τελευταίο στοιχείο τους και το βήμα:

```
» odds=1:2:9
odds =
     1     3     5     7     9
» evens=0:2:8
evens =
     0     2     4     6     8
```

Το βήμα μπορεί να παραληφθεί, όταν έχει την τιμή 1:

```
» first_10_numbers=1:10
first_10_numbers =
     1     2     3     4     5     6     7     8     9    10
```

όπως μπορεί να πάρει και αρνητικές ή κλασματικές τιμές:

```
» first_10_numbers_inverse=10:-1:1
first_10_numbers_inverse =
    10     9     8     7     6     5     4     3     2     1
» half_step=1:0.5:4
half_step =
    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
```

Μια χρήσιμη συνάρτηση είναι η `length`, η οποία επιστρέφει το πλήθος των στοιχείων μιας σειράς:

```
» length(odds)
ans =
     5
```

2.2 Πράξεις με μονοδιάστατες σειρές

Οι πράξεις της πρόσθεσης και της αφαίρεσης για τις σειρές αριθμών εκτελούνται με τη χρήση των γωνστών τελεστών:

```
» odds=1:2:9
odds =
     1     3     5     7     9
» evens=0:2:9
evens =
     0     2     4     6     8
» odds+evens
ans =
     1     5     9    13    17
```

Αντίθετα ο πολλαπλασιασμός, η διαίρεση και η ύψωση σε δύναμη καθενός από τα στοιχεία μιας σειράς γίνεται προσθέτοντας μια τελεία πριν από τον αντίστοιχο τελεστή:

```
» odds.*evens

ans =

     0     6    20    42    72

» evens.^2

ans =

     0     4    16    36    64
```

Ο πολλαπλασιασμός ή η διαίρεση όλων των στοιχείων με τον ίδιο αριθμό δεν απαιτούν τη χρήση της τελείας:

```
» 3*odds

ans =

     3     9    15    21    27

» evens/2

ans =

     0     1     2     3     4
```

Ένα από τα βασικά πλεονεκτήματα του MATLAB είναι ότι πολλές από τις ενσωματωμένες συναρτήσεις του υποστηρίζουν τη χρήση σειρών στη θέση των ορισμάτων με απλή αναφορά του ονόματος της σειράς. Αν θέλουμε, για παράδειγμα των υπολογισμό του ημιτόνου των γωνιών από 0° ως 180° ανά 10°, τότε μπορούμε να γράψουμε:

```
» gwnies=0:10:180;
» gwnies=gwnies*pi/180;
» sin_gwnies=sin(gwnies)

sin_gwnies =

  Columns 1 through 7

         0    0.1736    0.3420    0.5000    0.6428    0.7660    0.8660

  Columns 8 through 14

    0.9397    0.9848    1.0000    0.9848    0.9397    0.8660    0.7660

  Columns 15 through 19

    0.6428    0.5000    0.3420    0.1736    0.0000
```

2.2.1 Διανύσματα γραμμής και στήλης

Οι σειρές του MATLAB, όπως περιγράφηκαν παραπάνω, μπορούν να αναπαραστήσουν διανύσματα γραμμής της γραμμικής άλγεβρας:

$$A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$$

Με το MATLAB είναι δυνατή η αναπαράσταση διανυσμάτων και με τη μορφή στήλης, δηλαδή

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

τα οποία εισάγονται, αν αντί για το κόμμα χρησιμοποιήσουμε το ελληνικό ερωτηματικό (semi-colon) μεταξύ των στοιχείων μιας σειράς:

```
» odds=[1;3;5;7;9]

odds =

     1
     3
     5
     7
     9
```

Ένα διάνυσμα γραμμής μπορεί να μετατραπεί σε διάνυσμα στήλης και αντίστροφα, χρησιμοποιώντας το μοναδιαίο τελεστή της αναστροφής (transpose), ο οποίος συμβολίζεται με την απόστροφο ‘ στο MATLAB:

```
» odds=1:2:9

odds =

     1     3     5     7     9

» oddsT=odds'

oddsT =

     1
     3
     5
     7
     9
```

Με τη βοήθεια των διανυσμάτων στήλης και γραμμής μπορούμε να υπολογίσουμε εύκολα και το εσωτερικό (αριθμητικό) γινόμενο δύο διανυσμάτων $A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$ και $B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ ως $A * B = \sum_i^n a_i b_i$. Πράγματι, χρησιμοποιώντας τον τελεστή του πολλαπλασιασμού * σε αυτήν την περίπτωση, μπορούμε να βρούμε το εσωτερικό γινόμενο. Ως παράδειγμα θα υπολογίσουμε το άθροισμα των τετραγώνων των πρώτων πέντε φυσικών αριθμών:

```
» first_five=1:5;
» first_five*first_five'

ans =

    55
```

Θα επαληθεύσουμε το αποτέλεσμα με τη χρήση της συνάρτησης `sum`, η οποία μπορεί να υπολογίζει το άθροισμα των στοιχείων μιας μονοδιάστασης σειράς:

```
» first_five=1:5;
» sum(first_five.^2)

ans =

    55
```

2.2.2 Απλές γραφικές παραστάσεις

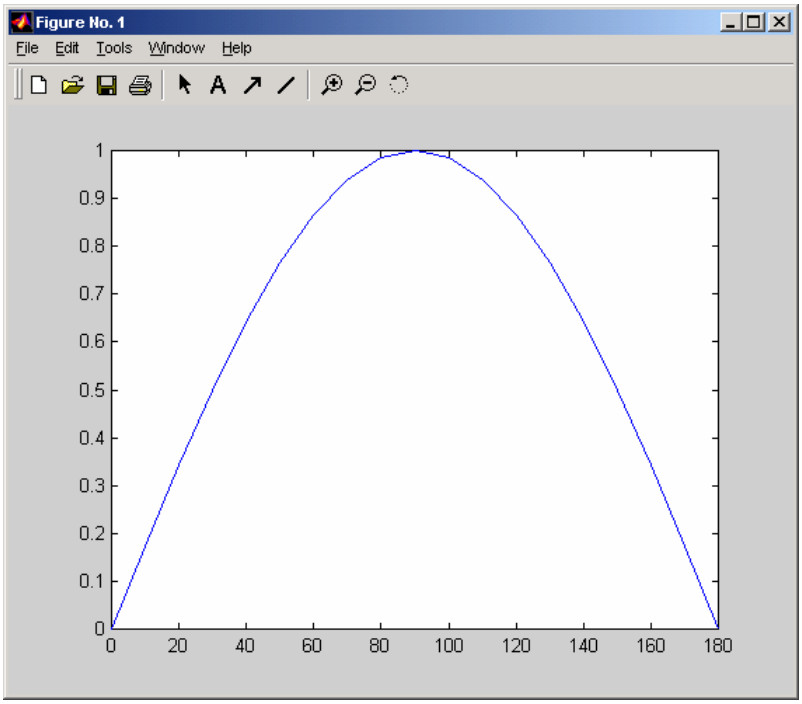
Έχοντας υπολογίσει παραπάνω τα ημίτονα της σειράς των γωνιών μπορούμε να κατασκευάσουμε την καμπύλη του ημιτόνου από 0 ως π με την εντολή `plot`. Η εντολή αυτή μπορεί να δημιουργήσει τη γραφική παράσταση δύο (ή περισσότερων) διανυσμάτων γραμμής ή δύο διανυσμάτων στήλης με το ίδιο πλήθος στοιχείων. Πληκτρολογώντας στη γραμμή εντολής:

```
» x=0:10:180;
» y=sin(x*pi/180);
» plot(x,y)
```

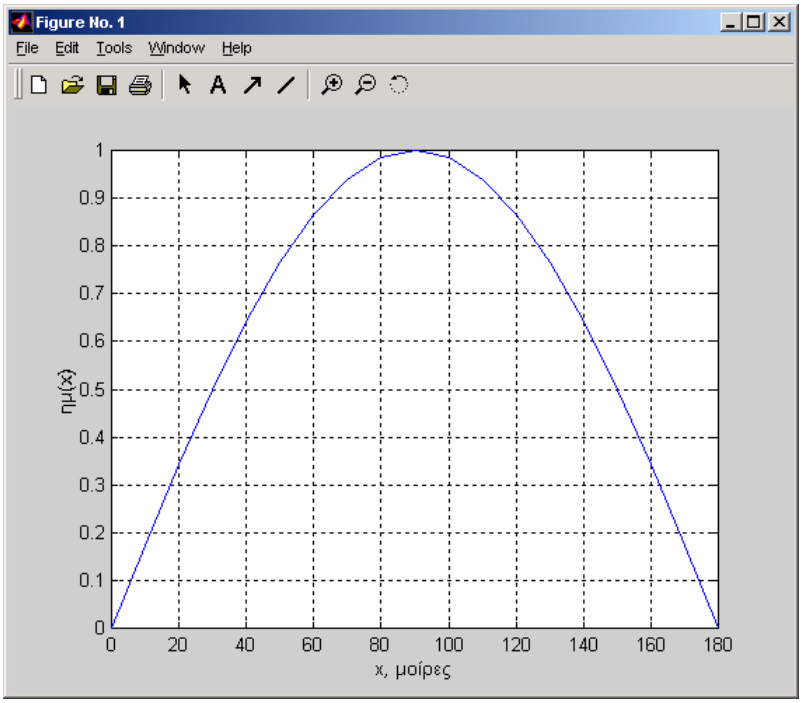
ανοίγει ένα καινούργιο παράθυρο που περιέχει τη ζητούμενη γραφική παράσταση (Σχήμα 1). Αν πληκτρολογήσουμε

```
» grid
» xlabel('x, μοίρες')
» ylabel('ημ(x)')
```

παίρνουμε την εικόνα του Σχήματος 2. Η εντολή `grid` εμφανίζει (και εξαφανίζει) το πλέγμα της γραφικής παράστασης. Οι εντολές `xlabel` και `ylabel` δέχονται ως όρισμα μια ακολουθία χαρακτήρων (string), η οποία χρησιμοποιείται ως τίτλος του οριζόντιου και του κατακόρυφου άξονα της γραφικής παράστασης, αντίστοιχα.



Σχήμα 1. Γραφική παράσταση ημιτόνου.



Σχήμα 2. Τελική μορφή της γραφικής παράστασης του ημιτόνου.

3 Πίνακες

3.1 Ορισμός

Οι πίνακες είναι δισδιάστατες σειρές, δηλαδή διατεταγμένα σύνολα αριθμών που προσδιορίζονται από δύο δείκτες. Οι πίνακες εισάγονται με τρόπο αντίστοιχο με αυτόν των μονοδιάστατων σειρών, χρησιμοποιώντας κόμμα ή κενό για την εισαγωγή στοιχείων της ίδιας γραμμής και το ελληνικό ερωτηματικό για να δηλώσουμε την αλλαγή της γραμμής:

```
» A=[1,2,3 ; 4 5 6]

A =

     1     2     3
     4     5     6
```

Πρέπει να τονιστεί ότι το πλήθος των στοιχείων κάθε γραμμής πρέπει να είναι το ίδιο, αλλιώς εμφανίζεται μήνυμα λάθους:

```
» A=[1,2,3 ; 4 5 ]
??? A=[1,2,3 ; 4 5 ]
                  |
All rows in the bracketed expression must have the same
number of columns.
```

Στους πίνακες χρησιμοποιούνται δύο δείκτες για τον προσδιορισμό του κάθε στοιχείου, κατά τον ίδιο τρόπο που χρησιμοποιούνται οι δείκτες στην περίπτωση των μονοδιάστατων σειρών, δηλαδή και για εμφάνιση αλλά και για εισαγωγή των στοιχείων ενός πίνακα:

```
» A(2,3)

ans =

     6
```

Οι διαστάσεις ενός πίνακα λαμβάνονται με τη συνάρτηση `size`:

```
» size(A)

ans =

     2     3
```

Μπορούμε να δημιουργήσουμε πίνακες μεγαλύτερων διαστάσεων με παράθεση, δηλαδή τοποθετώντας δύο (ή περισσότερους) πίνακες τον έναν δίπλα στον άλλο με κενό ή κόμμα (αρκεί να έχουν τον ίδιο αριθμό γραμμών) ή τον ένα κάτω από τον άλλο με χρήση του ελληνικού ερωτηματικού (οπότε πρέπει να έχουν τον ίδιο αριθμό στηλών). Για παράδειγμα, αν δημιουργήσουμε τους πίνακες:

```
A =

     1     2     3
     4     5     6

» A=[1 2 3 ; 4 5 6]

A =

     1     2     3
     4     5     6

» B=[11 12 13 ; 14 15 16]

B =

    11    12    13
    14    15    16
```

τότε μπορούμε να γράψουμε και

```
» C=[A B]

C =

     1     2     3    11    12    13
     4     5     6    14    15    16

» C=[A;B]

C =

     1     2     3
     4     5     6
    11    12    13
    14    15    16
```

Επίσης, είναι δυνατόν να εξάγουμε έναν υποπίνακα από έναν πίνακα. Από τον τελευταίο πίνακα, έστω ότι θέλουμε να πάρουμε των υποπίνακα που αποτελείται από τα στοιχεία των τελευταίων δύο γραμμών στις τελευταίες δύο στήλες:

```
» C(3:4,2:3)

ans =

    12    13
    15    16
```

Αν θέλουμε να πάρουμε όλα τα στοιχεία της δεύτερης στήλης του πίνακα γράφουμε

```
» C(:,2)

ans =

     2
     5
    12
    15
```

ενώ για όλα τα στοιχεία της τρίτης γραμμής γράφουμε

```
» C(3,:)

ans =

    11    12    13
```

3.2 Πράξεις με πίνακες

Αν θεωρήσουμε τους πίνακες A και B που ορίσαμε στα προηγούμενα παραδείγματα, τότε μπορούμε να δείξουμε ότι οι πράξεις με πίνακες στο MATLAB γίνονται εύκολα, χρησιμοποιώντας τους γνωστούς τελεστές. Για την πρόσθεση και την αφαίρεση των δύο παραπάνω πινάκων με διαστάσεις 2×3 έχουμε

```
» C=A+B

C =

    12    14    16
    18    20    22

» C=B-A

C =

    10    10    10
    10    10    10
```

Αντίθετα, όπως γνωρίζουμε, στον πολλαπλασιασμό πρέπει να προσέξουμε τις διαστάσεις των πινάκων. Έτσι, ο πολλαπλασιασμός ενός πίνακα 2×3, όπως είναι ο A, με έναν πίνακα ίδιων διαστάσεων, όπως ο B, δεν είναι δυνατός. Μπορούμε όμως να πάρουμε έναν νέο πίνακα Bt, ως τον ανάστροφο του B, χρησιμοποιώντας και πάλι την απόστροφο, όπως στην περίπτωση των μονοδιάστατων διανυσμάτων:

```
» Bt=B'

Bt =

    11    14
    12    15
    13    16
```

Τώρα είναι δυνατός ο πολλαπλασιασμός του A με τον Bt, ο οποίος θα δώσει έναν πίνακα διαστάσεων 2×2:

```
» C=A*Bt

C =

    74    92
   182   227
```

Όπως και στην περίπτωση των μονοδιάστατων σειρών, είναι δυνατόν να πολλαπλασιάσουμε δυο πίνακες ίδιων διαστάσεων κατά στοιχείο, χρησιμοποιώντας την τελεία μπροστά από τον τελεστή του πολλαπλασιασμού, δηλαδή:

```
» C=A.*B

C =

    11    24    39
    56    75    96
```

Με παρόμοιο τρόπο μπορούμε να υψώσουμε στην ίδια δύναμη το στοιχείο κάθε πίνακα:

```
» C=A.^2

C =

     1     4     9
    16    25    36
```

Ο τελεστής της ύψωσης σε δύναμη μπορεί να χρησιμοποιηθεί χωρίς την τελεία, μόνο στην περίπτωση τετραγωνικού πίνακα (ίδιος αριθμός γραμμών και στηλών), γιατί μεταφράζεται ως πολλαπλασιασμός του πίνακα με τον εαυτό του:

```
» A=[1 2 ; 3 4]

A =

     1     2
     3     4

» A^2

ans =

     7    10
    15    22

» A^3

ans =

    37    54
    81   118
```

Το MATLAB έχει το πλεονέκτημα ότι πολλές από τις ενσωματωμένες συναρτήσεις του δέχονται ως όρισμα πίνακες:

```
» A=[1/2 1/3 ; 1/4 1/6]

A =

    0.5000    0.3333
    0.2500    0.1667

» A=pi*A

A =

    1.5708    1.0472
    0.7854    0.5236

» sin(A)

ans =

    1.0000    0.8660
    0.7071    0.5000
```

Δυο πολύ χρήσιμες εντολές είναι οι `ones` και `zeros`, οι οποίες χρησιμοποιούνται για τη δημιουργία μοναδιαίων και μηδενικών πινάκων, αντίστοιχα, με τη χρήση δύο ορισμάτων, από τα οποία το πρώτο αντιστοιχεί στο πλήθος των γραμμών και το δεύτερο στο πλήθος των στηλών:

```
» ones(2,2)

ans =

     1     1
     1     1

» zeros(2,3)

ans =

     0     0     0
     0     0     0
```

3.3 Εμφάνιση με μορφή πίνακα

Σε προηγούμενο παράδειγμα υπολογίσαμε το ημίτονο των γωνιών από 0° ως 180° με βήμα 10°. Μπορούμε να εμφανίσουμε το αποτέλεσμα ως πίνακα, χρησιμοποιώντας τις λειτουργίες της αναστροφής και της παράθεσης:

```
» gwnies=0:10:180;
» gwnies=gwnies*pi/180;
```

```
» sin_gwnies=sin(gwnies);
» [gwnies' sin_gwnies']

ans =

         0         0
    0.1745    0.1736
    0.3491    0.3420
.....................
    2.9671    0.1736
    3.1416    0.0000
```

4 Μιγαδικοί αριθμοί

4.1 Στοιχειώδεις πράξεις με μιγαδικούς αριθμούς

Ο ορισμός και η χρησιμποίηση μιγαδικών μεταβλητών γίνονται πολύ απλά στο MATLAB. Η μιγαδική μονάδα είναι σταθερά του MATLAB αποθηκευμένη με το όνομα i και j:

```
» i

ans =

        0 + 1.0000i

» j

ans =

        0 + 1.0000i
```

Συνήθως τα ονόματα αυτών των μεταβλητών χρησιμοποιούνται και στους βρόχους επανάληψης ως μετρητές (όπως θα δούμε παρακάτω). Επομένως, χρειάζεται προσοχή, γιατί, αν οι μεταβλητές i και j ξαναοριστούν από τον χρήστη, παύουν να συμβολίζουν τη μιγαδική μονάδα. Πάντως στην εμφάνιση της απάντησης (αποτελέσματος υπολογισμών) το MATLAB χρησιμοποιεί πάντα τη μεταβλητή i για το συμβολισμό της μιγαδικής μονάδας.

Η εισαγωγή μιγαδικών μεταβλητών γίνεται ως εξής:

```
» z1=2+3i

z1 =

   2.0000 + 3.0000i

» z2=5+2j

z2 =
```

```
   5.0000 + 2.0000i
```

Οι πράξεις με μιγαδικούς αριθμούς γίνονται με τον ίδιο τρόπο, όπως και για τους πραγματικούς αριθμούς:

```
» z3=z1/z2
z3 =
   0.5517 + 0.3793i
```

Με τις εντολές `real` και `imag` μπορούμε να πάρουμε το πραγματικό και το φανταστικό μέρος ενός μιγαδικού αριθμού, αντίστοιχα:

```
» real(z1)
ans =
     2
» imag(z1)
ans =
     3
```

Με την εντολή `conj` παίρνουμε το μιγαδικό συζυγή. Με την εντολή `abs` παίρνουμε το μέτρο και με την εντολή `angle` το όρισμα ενός μιγαδικού αριθμού (σε ακτίνια):

```
» conj(z1)
ans =
   2.0000 - 3.0000i
» abs(z1)
ans =
    3.6056
» sqrt(z1*conj(z1))
ans =
    3.6056
```

```
» angle(z1)

ans =

    0.9828
```

4.2 Σχεδίαση μιγαδικού αριθμού

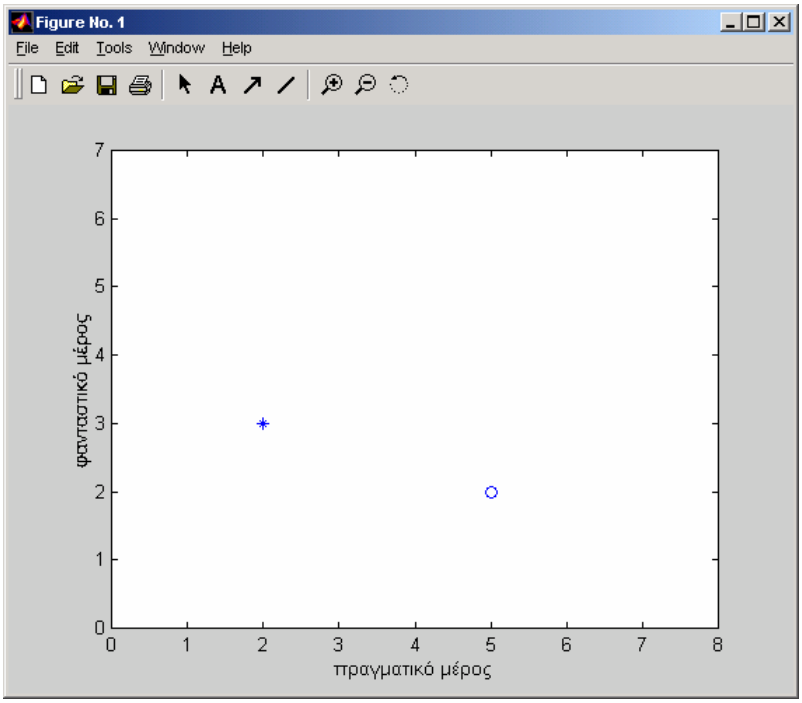
Τους μιγαδικούς αριθμούς μπορούμε να τους σχεδιάσουμε στο μιγαδικό επίπεδο, χρησιμοποιώντας την εντολή `plot`. Για παράδειγμα, με τις παρακάτω γραμμές στο σχήμα που προκύπτει (Σχήμα 3) αναπαρίστανται δύο μιγαδικοί αριθμοί, ο ένας με αστερίσκο και ο άλλος με έναν κύκλο στο μιγαδικό επίπεδο:

```
» plot(z1,'*')
» hold
Current plot held
» plot(z2,'o')
» axis([0 8 0 7])
» xlabel('πραγματικό μέρος')
» ylabel('φανταστικό μέρος')
```

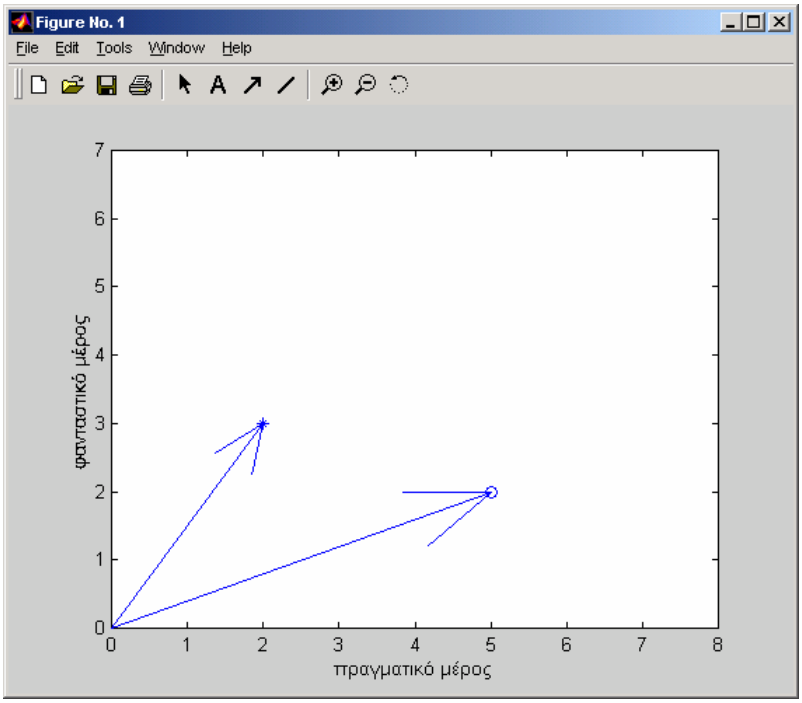
Το δεύτερο όρισμα της εντολής `plot` χρησιμοποείται ως το σύμβολο για το σχεδιασμό του αντίστοιχου μιγαδικού αριθμού. Η εντολή `hold` επιτρέπει τη σχεδίαση των δυο αριθμών στους ίδιους άξονες, ενώ η εντολή `axis([0 8 0 7])` αλλάζει τα άνω και κάτω όρια του οριζόντιου και του κατακόρυφου άξονα. Προσθέτοντας τις εντολές

```
» compass(z1)
» compass(z2)
```

παίρνουμε την αναπαράσταση των δυο μιγαδικών αριθμών (Σχήμα 4) και ως στρεφόμενα διανύσματα (phasors).



Σχήμα 3. Αναπαράσταση δύο μιγαδικών αριθμών στο μιγαδικό επίπεδο.



Σχήμα 4. Αναπαράσταση μιγαδικών αριθμών ως στρεφόμενα διανύσματα.

4.3 Σειρές και πίνακες μιγαδικών αριθμών

Ένα σημείο που αξίζει να αναφερθεί για τη χρήση σειρών και πινάκων μιγαδικών αριθμών είναι ότι η χρήση της αποστρόφου δεν κάνει απλή αναστροφή της σειράς ή του πίνακα, αλλά παράγει την ανάστροφο μιγαδική συζυγή σειρά ή τον ανάστροφο μιγαδικό συζυγή πίνακα:

```
» z=[1+2i 3+5i 1+4i]

z =

   1.0000 + 2.0000i   3.0000 + 5.0000i   1.0000 + 4.0000i

» z'

ans =

   1.0000 - 2.0000i
   3.0000 - 5.0000i
   1.0000 - 4.0000i
```

Για να πάρουμε απλά την ανάστροφο σειρά ή τον ανάστροφο πίνακα, πρέπει να χρησιμοποιήσουμε την τελεία πριν από την απόστροφο:

```
» z=[1+2i 3+5i 1+4i]

z =

   1.0000 + 2.0000i   3.0000 + 5.0000i   1.0000 + 4.0000i

» z.'

ans =

   1.0000 + 2.0000i
   3.0000 + 5.0000i
   1.0000 + 4.0000i
```

5 Συστήματα γραμμικών εξισώσεων

5.1 Ορίζουσες

Ας θεωρήσουμε το ακόλουθο σύστημα γραμμικών εξισώσεων:

$$\begin{aligned} x_1 + x_2 + 4x_3 &= 7 \\ x_1 - 2x_2 + 6x_3 &= 15 \\ 2x_1 + x_2 - x_3 &= 7 \end{aligned}$$

Ο πίνακας

$$A = \begin{bmatrix} 1 & 1 & 4 \\ 1 & -2 & 6 \\ 2 & 1 & -1 \end{bmatrix}$$

ορίζεται ως πίνακας συντελεστών του συστήματος και το διάνυσμα στήλης

$$B=\begin{bmatrix} 7 \\ 15 \\ 7 \end{bmatrix}$$

ως το δεξιό μέρος (σταθερών όρων) του συστήματος. Αν θεωρήσουμε ως λύση του συστήματος το διάνυσμα στήλης

$$X=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

τότε το σύστημα των γραμμικών εξισώσεων γράφεται

$$AX = B$$

Με πολύ εύκολο τρόπο οι πίνακες A και B εισάγονται στο MATLAB:

```
» A=[1 1 4; 1 -2 6; 2 1 -1]

A =

     1     1     4
     1    -2     6
     2     1    -1

» B=[7; 15; 7]

B =

     7
    15
     7
```

Ένας τρόπος επίλυσης του συστήματος αυτού είναι με τον κανόνα του Cramer. Αν και δεν πρόκειται για τον πλέον αποτελεσματικό τρόπο, ιδιαίτερα όταν το σύστημα των εξισώσεων είναι μεγάλο, θα εφαρμοστεί εδώ, για να δειχθεί η ευκολία με την οποία υπολογίζονται οι ορίζουσες πινάκων στο MATLAB. Για παράδειγμα ο υπολογισμός της ορίζουσας του πίνακα A γίνεται, γράφοντας απλά:

```
» det(A)

ans =

    29
```

5.2 Κανόνας του Cramer

Είναι γνωστό ότι, για να βρεθεί η λύση του παραπάνω συστήματος με τον κανόνα του Cramer, αντικαθίστανται διαδοχικά οι τρεις στήλες του συντελεστών A με το διάνυσμα στήλη B των σταθερών όρων. Υπολογίζεται σε κάθε περίπτωση η ορίζουσα και διαιρείται με την ορίζουσα του πίνακα συντελεστών. Η εφαρμογή του κανόνα Cramer για το παραπάνω σύστημα στο MATLAB γίνεται ως εξής:

```
» DX1=A; DX1(:,1)=B

DX1 =

     7     1     4
    15    -2     6
     7     1    -1

» X1=det(DX1)/det(A)

X1 =

     5

» DX2=A; DX2(:,2)=B

DX2 =

     1     7     4
     1    15     6
     2     7    -1

» X2=det(DX2)/det(A)

X2 =

    -2

» DX3=A; DX3(:,3)=B

DX3 =

     1     1     7
     1    -2    15
     2     1     7

» X3=det(DX3)/det(A)

X3 =

     1

» X=[X1; X2; X3]

X =
```

```
     5
    -2
     1
```

Η επαλήθευση του αποτελέσματος γίνεται με τον πολλαπλασιασμό:

```
» A*X

ans =

     7
    15
     7
```

5.3 Αντιστροφή πινάκων

Αφού η ορίζουσα του πίνακα συντελεστών είναι διάφορη του μηδενός, μπορεί να οριστεί ο αντίστροφος πίνακας του πίνακα συντελεστών , τον οποίο το MATLAB υπολογογίζει ως εξής:

```
» A_inverse=inv(A)

A_inverse =

   -0.1379    0.1724    0.4828
    0.4483   -0.3103   -0.0690
    0.1724    0.0345   -0.1034
```

Στην περίπτωση αυτή μπορούμε να βρούμε εύκολα τη λύση του συστήματος ως:

```
» X=A_inverse*B

X =

    5.0000
   -2.0000
    1.0000
```

5.4 Άμεση μέθοδος επίλυσης

Η πιο απλή μέθοδος άμεσης επίλυσης ενός συστήματος είναι αυτή της απαλοιφής Gauss. Περιλαμβάνει δύο μέρη: (α) τη διαδικασία της απαλοιφής ή τριγωνισμού, και (β) τη διαδικασία της αντικατάστασης. Χρησιμοποιώντας το παραπάνω σύστημα γραμμικών αλγεβρικών εξισώσεων ως παράδειγμα, η απαλοιφή γίνεται ως εξής: Αφαιρούμε την πρώτη εξίσωση από

τη δεύτερη και τη πρώτη πολλαπλασιασμένη με δυο από την τρίτη. Με τον τρόπο αυτό προκύπτει ένα σύστημα, στο οποίο ο άγνωστος x_1 έχει απαλειφθεί από όλες τις εξισώσεις εκτός από την πρώτη:

$$\begin{aligned} x_1 + x_2 + 4x_3 &= 7 \\ -3x_2 + 2x_3 &= 8 \\ -x_2 - 9x_3 &= -7 \end{aligned}$$

Ο συντελεστής του αγνώστου x_1 στην πρώτη εξίσωση ονομάζεται άξονας (pivot) και καθορίζει τους κατάλληλους πολλαπλασιαστές της πρώτης εξίσωσης, με τους οποίους αυτή πρέπει να πολλαπλασιαστεί πριν αφαιρεθεί από τις επόμενες εξισώσεις (1 και 2 για τη δεύτερη και τρίτη εξίσωση αντίστοιχα).

Τώρα προχωρούμε στην απαλοιφή του αγνώστου x_2 από την τρίτη εξίσωση του τροποποιημένου συστήματος. Ο άξονας είναι πλέον ο αριθμός -3 (δηλαδή ο συντελεστής του x_2). Αφαιρώντας τη δεύτερη εξίσωση πολλαπλασιασμένη με 1/3 από την τρίτη παίρνουμε το σύστημα

$$\begin{aligned} x_1 + x_2 + 4x_3 &= 7 \\ -3x_2 + 2x_3 &= 8 \\ -\frac{29}{3}x_3 &= -\frac{29}{3} \end{aligned}$$

Ο πίνακας που αντιστοιχεί σε αυτό το σύστημα είναι άνω τριγωνικός (όλα τα στοιχεία κάτω από την κύρια διαγώνιό του είναι μηδενικά). Το σύστημα επιλύεται τώρα πολύ εύκολα. Από την τρίτη εξίσωση προκύπτει άμεσα ότι $x_3 = 1$. Αντικαθιστώντας στη δεύτερη εξίσωση παίρνουμε ότι $x_2 = -2$ και, τελικά, με αντικάτασταση των τιμών για τους x_2 και x_3 στην πρώτη εξίσωση προκύπτει ότι $x_1 = 5$.

Όλη η παραπάνω διαδικασία μπορεί να ενεργοποιηθεί στο MATLAB με τον τελεστή \ ως εξής:

```
» X=A\B

X =

    5.0000
   -2.0000
    1.0000
```

Στην παραπάνω διαδικασία πρέπει να προβλεφθεί και η περίπτωση του μηδενικού άξονα. Στην περίπτωση αυτή δυο εξισώσεις εναλλάσσονται, ώστε να μην υπάρχει μηδενικός

άξονας. Αυτή η εναλλαγή είναι πάντα δυνατή αν ο πίνακας A είναι αντιστρέψιμος (μη-ιδιάζων). Μια σημαντική αριθμητική αστάθεια μπορεί να προκύψει, αν ο άξονας είναι πολύ μικρός σε σχέση με τους υπόλοιπους συντελεστές στη στήλη του. Στην περίπτωση αυτή και η πιο μικρή ανακρίβεια που υπήρχε αρχικά είτε στα δεδομένα του πίνακα A είτε στο διάνυσμα στήλης B θα αυξηθεί, όπως θα αυξηθούν και τα σφάλματα στρογγυλοποίησης. Αυτό οφείλεται στην πεπερασμένη ακρίβεια αποθήκευσης των αριθμών στους ηλεκτρονικούς υπολογιστές.

Ας υποθέσουμε ότι ένας ηλεκτρονικός υπολογιστής μπορεί να αποθηκεύσει επτά σημαντικά δεκαδικά ψηφία για αριθμούς κινητής υποδιαστολής. Θεωρούμε το σύστημα γραμμικών αλγεβρικών εξισώσεων

$$\begin{aligned} 1.000000\times10^{-8}\ \mathrm{x}_1 + 1.000000\ \mathrm{x}_2 &= 1.000000 \\ 1.000000\ \mathrm{x}_1 + 1.000000\ \mathrm{x}_2 &= 2.000000 \end{aligned}$$

Αν επιχειρήσουμε απαλοιφή Gauss, χωρίς άξονες, και λάβουμε υπόψη τον περιορισμό στην ακρίβεια αποθήκευσης των αριθμών κινητής υποδιαστολής, τότε παίρνουμε το τροποποιημένο σύστημα

$$\begin{aligned} 1.000000\times10^{-8}\ \mathrm{x}_1 + 1.000000\ \mathrm{x}_2 &= 1.000000 \\ -1.000000\times10^{8}\ \mathrm{x}_2 &= -1.000000\times10^{8} \end{aligned}$$

Η αντικατάσταση προς τα πίσω δίνει $\mathrm{x}_2 = 1.000000$ και $\mathrm{x}_1 = 0.000000$, λύση η οποία είναι προφανώς λανθασμένη. Αντιστρέφοντας τη σειρά των εξισώσεων και ακολουθώντας την ίδια διαδικασία παίρνουμε $\mathrm{x}_1 = 1.000000$ και $\mathrm{x}_2 = 1.000000$. (Σημείωση: Αν οι συντελεστές θεωρηθούν ότι αποθηκεύονται με απεριόριστη ακρίβεια στον υπολογιστή, τότε η ακριβής λύση είναι $\mathrm{x}_1 = 1+\frac{1}{10^8-1}$ και $\mathrm{x}_2 = 1-\frac{1}{10^8-1}$.) Ο γενικός κανόνας που προκύπτει από το παραπάνω παράδειγμα είναι ότι η σειρά των εξισώσεων πρέπει να αλλάζει, ώστε να λαμβάνεται ο άξονας με τη μεγαλύτερη τιμή.

Είναι προφανές ότι η περιορισμένη ακρίβεια με την οποία αποθηκεύονται οι αριθμοί κινητής υποδιαστολής στον ηλεκτρονικό υπολογιστή θέτει ένα όριο και στην ακρίβεια μιας λύσης που υπολογίζεται με την παραπάνω μέθοδο. Δυστυχώς για μεγάλα συστήματα εξισώσεων η συγκέντρωση των σφαλμάτων στρογγυλοποίησης μπορεί να οδηγήσει στην απώλεια μερικών ακόμα ψηφίων ακρίβειας.

5.5 Επαναληπτικές μέθοδοι επίλυσης

Η εφαρμογή της άμεσης μεθόδου επίλυσης σε φυσικά προβλήματα έδειξε ότι, καθώς το μέγεθος του πίνακα A αυξάνεται, ο χρόνος που απαιτείται για τον υπολογισμό της λύσης

ξεπερνά κατά πολύ το χρόνο που απαιτείται για το σχηματισμό του ίδιου του πίνακα. Στην περίπτωση αυτή αξίζει να εξετάσουμε τις επαναληπτικές μεθόδους. Για πυκνούς πίνακες που προκύπτουν από την εφαρμογή της μεθόδου των ροπών στον υπολογιστικό ηλεκτρομαγνητισμό η τεχνική της συζυγούς βαθμίδας (conjugate gradient) είναι ίσως η πλέον κατάλληλη. Αντίθετα για τους πίνακες που προκύπτουν από την εφαρμογή της μεθόδου πεπερασμένων στοιχείων, οι γραμμικές επαναληπτικές τεχνικές, όπως η τεχνική της διαδοχικής υπερχαλάρωσης (SOR: Successive Over-Relaxation) είναι καταλληλότερες. Στην παρούσα ενότητα παρουσιάζονται συνοπτικά αυτές οι τεχνικές.

5.5.1 Μέθοδος Jacobi

Θεωρούμε ότι έχουμε το εξής σύστημα

$$\begin{aligned} 4x_1 &- x_2 - x_3 = 5 \\ -x_1 &+ 4x_2 - x_3 = 0 \\ -x_1 &- x_2 - 4x_3 = 3 \end{aligned}$$

το οποίο μπορεί να γραφεί και ως

$$x_1 = \frac{1}{4}\left(5 + x_2 + x_3\right)$$

$$x_2 = \frac{1}{4}\left(0 + x_1 + x_3\right)$$

$$x_3 = \frac{1}{4}\left(3 + x_1 + x_2\right)$$

Υποθέτουμε ότι $x^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)$ είναι μια προσεγγιστική λύση του συστήματος. Ένας λογικός τρόπος για να σχηματίσουμε μια καινούρια προσεγγιστική λύση είναι να υπολογίσουμε τα

$$x_1^{(k+1)} = \frac{1}{4}\left(5 + x_2^{(k)} + x_3^{(k)}\right)$$

$$x_2^{(k+1)} = \frac{1}{4}\left(0 + x_1^{(k)} + x_3^{(k)}\right)$$

$$x_3^{(k+1)} = \frac{1}{4}\left(3 + x_1^{(k)} + x_2^{(k)}\right)$$

Γενικά για ένα σύστημα N × N που περιγράφεται από την $Ax = B$, αυτού του είδους η επανάληψη ονομάζεται “επανάληψη Jacobi” και μπορεί να αναπαρασταθεί, αν γράψουμε τον

A ως άθροισμα ενός πίνακα D με τα διαγώνια στοιχεία του και ενός πίνακα $\left(A - D\right)$ με τα υπόλοιπα στοιχεία του, όπου

$$D = \begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

Το αρχικό σύστημα εξισώσεων γράφεται τότε ως

$$D\,x = B - \left(A - D\right)x$$

Με την προϋπόθεση ότι κανένα από τα διαγώνια στοιχεία του δεν είναι μηδενικό, ο πίνακας D αντιστρέφεται πολύ εύκολα και προκύπτει το επαναληπτικό σχήμα

$$x^{(k+1)} = D^{-1}\left[B - \left(A - D\right)x^{(k)}\right]$$

Οι επαναλήψεις συνεχίζονται ώσπου η διαφορά μεταξύ δύο διαδοχικών λύσεων είναι σχετικά μικρή (μικρότερη από ένα όριο που έχουμε θέσει). Η σύγκλιση είναι εξασφαλισμένη, όταν κάθε διαγώνιο στοιχείο είναι μεγαλύτερο από το άθροισμα των απολύτων τιμών των υπόλοιπων στοιχείων στη γραμμή του.

Η εφαρμογή του παραπάνω αλγόριθμου στο MATLAB είναι σχετικά απλή. Με χρήση της εντολής `diag` σχηματίζουμε το διαγώνιο πίνακα D πριν αρχίσουμε τις επαναλήψεις:

```
» A=[4 -1 -1; -1 4 -1; -1 -1 -4]

A =

     4    -1    -1
    -1     4    -1
    -1    -1    -4

» B=[5; 0; 3]

B =

     5
     0
     3

» D=diag(diag(A))

D =

     4     0     0
     0     4     0
     0     0    -4
```

Ως αρχική λύση θεωρούμε την

```
» x=[0; 0; 0]

x =

     0
     0
     0
```

Η πρώτη επανάληψη γίνεται, γράφοντας:

```
» x=inv(D)*(B-(A-D)*x)

x =

   1.25000000000000
                   0
  -0.75000000000000
```

Μετά από εννέα επαναλήψεις παίρνουμε:

```
» x=inv(D)*(B-(A-D)*x)

x =

   0.99995803833008
  -0.00004577636719
  -1.00004196166992
```

Μπορούμε να συγκρίνουμε τη λύση αυτή με το αποτέλεσμα τη μεθόδου απαλοιφής Gauss:

```
» x=A\B

x =

     1
     0
    -1
```

5.5.2 Μέθοδος Gauss-Seidel

Η επανάληψη Jacobi μπορεί να τροποποιηθεί, ώστε σε κάθε επανάληψη να γίνεται χρήση της πλέον πρόσφατης τιμής για κάθε άγνωστο. Η μέθοδος που λειτουργεί με αυτόν τον τρόπο ονομάζεται “μέθοδος Gauss-Seidel” και για το σύστημα γραμμικών αλγεβρικών εξισώσεων του παραδείγματος λειτουργεί ως εξής

$$x_1^{(k+1)} = \frac{1}{4}\left(5 + x_2^{(k)} + x_3^{(k)}\right)$$

$$x_2^{(k+1)} = \frac{1}{4}\left(0 + x_1^{(k+1)} + x_3^{(k)}\right)$$

$$x_3^{(k+1)} = \frac{1}{4}\left(3 + x_1^{(k+1)} + x_2^{(k+1)}\right)$$

Για να γράψουμε τη μέθοδο Gauss-Seidel με τη μορφή πινάκων, γράφουμε τον πίνακα $\left(A - D\right)$ ως το άθροισμα ενός αυστηρά άνω τριγωνικού πίνακα U, ενός αυστηρά κάτω τριγωνικού πίνακα A και του διαγώνιου πίνακα D, οπότε $A = L + D + U$. Το σύστημα γράφεται τότε ως

$$\left(L + D\right)x = B - Ux$$

και η μέθοδος με την μορφή πινάκων δίνεται ως

$$\left(L + D\right)x^{(k+1)} = B - Ux^{(k)}$$

Γνωρίζοντας ότι ο πίνακας $L + D$ είναι κάτω τριγωνικός, μπορούμε να υπολογίσουμε την επόμενη λύση με τη μέθοδο της αντικατάστασης προς τα εμπρός. Στην ουσία υπολογίζουμε τη λύση

$$x^{(k+1)} = \left(L + D\right)^{-1}\left(B - Ux^{(k)}\right)$$

Κάθε επανάληψη έχει περίπου το ίδιο υπολογιστικό κόστος με μια επανάληψη Jacobi. Η μέθοδος Gauss-Seidel συγκλίνει για μια κατηγορία συστημάτων, τα οποία χαρακτηρίζονται από πραγματικούς θετικά ορισμένους πίνακες A ($x^t A\, x > 0$ για κάθε μη-μηδενικό x). Τέτοιοι πίνακες προκύπτουν για παράδειγμα από τη διακριτοποίηση της εξίσωσης Poisson σε μια ορθογώνια περιοχή.

5.5.3 Μέθοδος SOR

Το πλεονέκτημα της παραπάνω μεθόδου είναι η απλότητά της. Το μειονέκτημά της όμως είναι η αργή πολλές φορές σύγκλισή της. Μια τροποποίηση της μεθόδου γίνεται με τη χρήση

μιας πραγματικής "παραμέτρου χαλάρωσης" ω, με την οποία εισάγεται ένα βάρος στη συνεισφορά της προηγούμενης και της πλέον πρόσφατης τιμής για τον υπολογισμό της λύσης στην επόμενη επανάληψη:

$$x_1^{(k+1)} = \frac{\omega}{4}\left(5 + x_2^{(k)} + x_3^{(k)}\right) + \left(1-\omega\right)x_1^{(k)}$$

$$x_2^{(k+1)} = \frac{\omega}{4}\left(0 + x_1^{(k+1)} + x_3^{(k)}\right) + \left(1-\omega\right)x_2^{(k)}$$

$$x_3^{(k+1)} = \frac{\omega}{4}\left(3 + x_1^{(k+1)} + x_2^{(k+1)}\right) + \left(1-\omega\right)x_3^{(k)}$$

Για μερικά σημαντικά προβλήματα η τιμή της βέλτιστης τιμής για την παράμετρο ω είναι γνωστή (σε κάθε περίπτωση πρέπει να είναι μεταξύ 0 και 2), οπότε επιτυγχάνεται πολύ γρήγορη σύγκλιση. Το επαναληπτικό αυτό σχήμα ονομάζεται "μέθοδος υπερ-χαλάρωσης" και περιγράφεται με τη μορφή πινάκων ως

$$x^{(k+1)} = \left(D + \omega L\right)^{-1}\left\{\omega B + \left[\left(1-\omega\right)D - \omega U\right]x^{(k)}\right\}$$

Γενικά μπορεί να πει κανείς ότι η σύγκλιση της μεθόδου Gauss-Seidel είναι γρηγορότερη από αυτήν της μεθόδου Jacobi και ότι η μέθοδος υπερ-χαλάρωσης συγκλίνει γρηγορότερα και από τις δυο. Δυστυχώς είναι δύσκολο, αν όχι αδύνατο, να διαπιστωθεί αν οι πίνακες που προκύπτουν από την εφαρμογή της μεθόδου των ροπών στις ολοκληρωτικές εξισώσεις είναι κατάλληλοι για την εφαρμογή των τεχνικών αυτών, οπότε δεν είναι πάντα εγγυημένη η σύγκλιση.

5.5.4 Μέθοδος συζυγούς βαθμίδος

Στα συστήματα αλγεβρικών εξισώσεων που προκύπτουν κατά την εφαρμογή της μεθόδου των ροπών εφαρμόζεται και η "μέθοδος της συζυγούς βαθμίδας". Αρχικά υποθέτουμε ότι το σύστημα $A\,x = B$ έχει μια λύση x_o. Η λύση αυτή έχει ένα υπόλοιπο $r_o = B - Ax_o$ και μια "συζυγή κατεύθυνση" $p_o = A^{*t} r_o$. Η μέθοδος παράγει μια σειρά από N διαδοχικές συζυγείς κατευθύνσεις $p_o, \ldots, p_{N-1}$ και διαδοχικές προσεγγίσεις $x_o, \ldots, x_{N-1}$ στην πραγματική λύση. Σε κάθε επανάληψη η ακριβής λύση x διορθώνεται κατά μια ποσότητα $a_i p_i$. Η διαδικασία ολοκληρώνεται ακριβώς μετά από N βήματα με την εύρεση της ακριβούς λύσης (για απόλυτα ακριβή αριθμητική στον υπολογιστή). Όμως και νωρίτερα από τα N βήματα είναι δυνατό να δώσει αρκετά ακριβείς λύσεις.

Ο αλγόριθμος περιγράφεται με τις σχέσεις

$$r_o = B - Ax_o$$

$$p_o = A^{*t} r_o$$

$$a_i = \frac{\|r_i\|^2}{\|p_i\|^2}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i A p_i$$

$$p_{i+1} = A^{*t} r_{i+1} + \frac{\|r_{i+1}\|^2}{\|r_i\|^2} p_i$$

Τα μεγάλα συστήματα μπορούν να δημιουργήσουν προβλήματα, αφού τα σφάλματα στρογγυλοποίησης οδηγούν σε απώλεια της ορθογωνικότητας μεταξύ των υπολοίπων r_i και έτσι αναιρούν την ιδιότητα του πεπερασμένου αριθμού βημάτων. Επομένως, πρέπει να χρησιμοποιούνται κριτήρια σχετικής ακρίβειας για τον τερματισμό της διαδικασίας.

6 Πολυώνυμα

6.1 Ρίζες ενός πολυωνύμου

Η αναπαράσταση των πολυωνύμων στο MATLAB γίνεται με καταχώρηση των συντελεστών των όρων τους με φθίνουσα σειρά των δυνάμεων της μεταβλητής τους, δηλαδή με τη σειρά που συνήθως αυτά γράφονται. Για παράδειγμα το πολυώνυμο

$$x^2 + 4x + 3$$

αποθηκεύεται σε ένα διάνυσμα-γραμμή του MATLAB ως

```
» polywnymo=[ 1  4  3 ];
```

Μπορούμε να υπολογίσουμε τις ρίζες του πολυωνύμου με τη χρήση της εντολής `roots`, ως εξής:

```
» r=roots(polywnymo)

r =

    -3
    -1
```

6.2 Σχεδίαση πολυωνύμου

Ας υποθέσουμε ότι θέλουμε να σχεδιάσουμε τη συνάρτηση

$$y = x^3 - 2x^2 - 5x + 6$$

σε ένα διάστημα τιμών που περιλαμβάνει τους μηδενισμούς της συνάρτησης. Πρώτα καταχωρούμε τους συντελεστές του πολυωνύμου:

```
» p=[1 -2 -5 6];
```

και βρίσκουμε τις ρίζες του:

```
» r=roots(p)

r =

   3.00000000000000
  -2.00000000000000
   1.00000000000000
```

Επομένως, αρκεί να σχεδιάσουμε τη συνάρτηση στο διάστημα [-3, 4]. Αυτό το κάνουμε ορίζοντας μια σειρά αριθμών με βήμα 0.2 σε αυτό το διάστημα και υπολογίζοντας τις τιμές της συνάρτησης για αυτή τη σειρά με την εντολή `polyval`, η οποία ως πρώτο όρισμα δέχεται το διάνυσμα με τους συντελεστές του πολυωνύμου και ως δεύτερο όρισμα το διάνυσμα με τη σειρά των αριθμών:

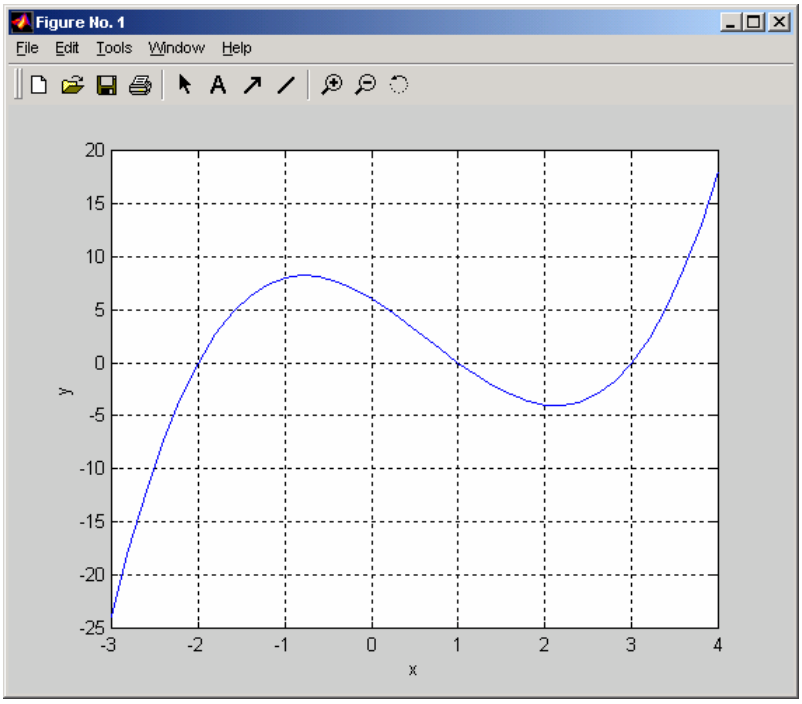
```
» x=-3:0.2:4;
» y=polyval(p,x);
```

Για τη σχεδίαση χρησιμοποιούμε την εντολή `plot`, οπότε προκύπτει η εικόνα του Σχήματος 5.

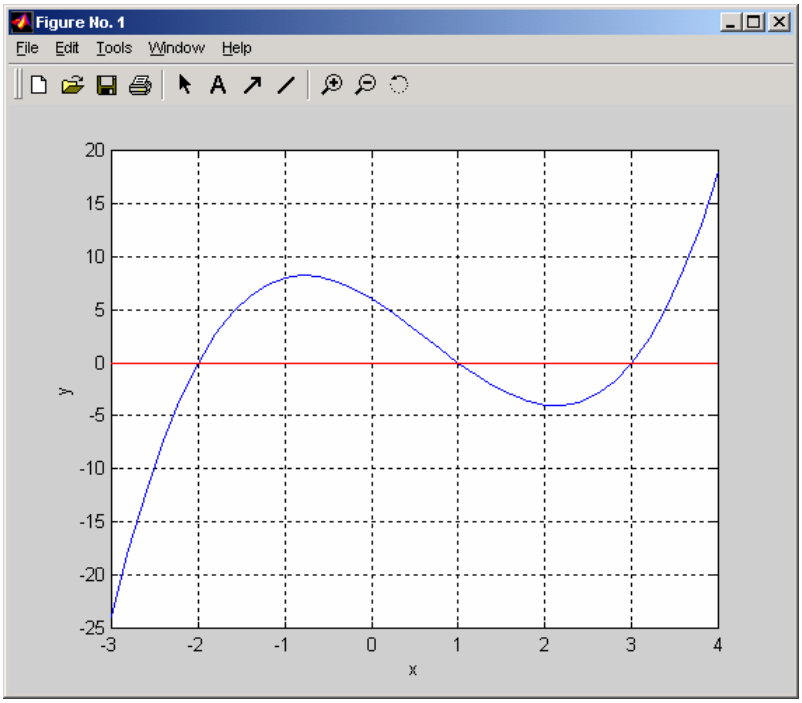
```
» plot(x,y)
» grid
» xlabel('x')
» ylabel('y')
```

Για να τονίσουμε τη οριζόντια γραμμή που τέμνει τον κατακόρυφο άξονα στο σημείο 0 (Σχήμα 6), σχεδιάζουμε μια ευθεία γραμμή κόκκινου χρώματος με τις εντολές:

```
» hold
Current plot held
» plot([-3,4],[0,0],'r')
```



Σχήμα 5. Σχεδίαση πολυωνυμικής συνάρτησης.



Σχήμα 6. Τελική σχεδίαση πολυωνυμικής συνάρτησης.

7 Προγραμματισμός στο MATLAB

7.1 Αρχείο κειμένου

Αντί για τη διαδοχική εκτέλεση των εντολών στη γραμμή εντολών, μπορούμε στο MATLAB να δημιουργήσουμε τα λεγόμενα m-αρχεία, τα οποία παραθέτουν τις εντολές που θέλουμε να εκτελεστούν. Τα αρχεία αυτά είναι αρχεία ASCII (text) και έχουν ως επίθεμα (extension) το '.m'. Οι εντολές που περιέχονται σε αυτά εκτελούνται με την αναγραφή στη γραμμή εντολών του MATLAB του ονόματος του αρχείου (χωρίς το επίθεμα). Η απλούστερη μορφή m-αρχείων είναι τα αρχεία κειμένου (script files). Θα πρέπει το m-αρχείο να βρίσκεται στο τρέχον μονοπάτι (path), για να μπορεί να εκτελεστεί. Έστω ότι δημιουργούμε ένα αρχείο με το όνομα 'hello.m' το οποίο σώζεται στον υποκατάλογο 'c:\temp' και περιέχει τις ακόλουθες γραμμές κώδικα:

```
% The program in this file greets you and asks for your
% name. Then it greets you again by your name.
disp('Hello! Who are you?')
name = input('Please enter your name enclosed between quotes: ');
answer = [ 'Hello ' name '!'];
disp(answer)
```

Η εκτέλεση του προγράμματος αυτού γίνεται με τις γραμμές:

```
» cd c:\temp
» hello
```

στη γραμμή εντολών του MATLAB. Με την εντολή `cd` (change directory) κάνουμε τον κατάλογο `c:\temp` τρέχοντα, ώστε να μπορέσουμε να καλέσουμε το αρχείο. Το αρχείο μας χαιρετά και ζητάει να δώσουμε το όνομά μας μέσα σε απλά εισαγωγικά. Στο τέλος χρησιμοποιεί την πληροφορία αυτή, για να μας χαιρετήσει με το όνομά μας:

```
Hello! Who are you?
Please enter your name enclosed between quotes: 'student'
Hello student!
```

Ας δούμε τώρα πιο αναλυτικά το πρόγραμμα. Οι δύο πρώτες γραμμές του προγράμματος αποτελούν σχόλια και αγνοούνται κατά την εκτέλεση του κώδικα. Τα σχόλια στα m-αρχεία του MATLAB εισάγονται με το χαρακτήρα % σε οποιοδήποτε σημείο του κώδικα. Αν πριν από τις πρώτες γραμμές σχολίων δεν υπάρχουν άλλες εντολές, τότε αυτά τα σχόλια μπορούν να εμφανιστούν με την εντολή `help`, δηλαδή αποτελούν ένα είδος επεξηγήσεως για τη

λειτουργία του αρχείου (πολύ καλή πρακτική, όταν προγραμματίζουμε σε MATLAB). Για παράδειγμα:

```
» help hello

  The program in this file greets you and asks for your
  name. Then it greets you again by your name.
```

Η εντολή `disp` εμφανίζει στην οθόνη μια ακολουθία χαρακτήρων μεταξύ απλών εισαγωγικών (single quotes). Στην επόμενη γραμμή ζητείται από το χρήστη το όνομά του, το οποίο αποθηκεύεται στη μεταβλητή `name`. Το περιεχόμενο της μεταβλητής αυτής χρησιμοποιείται, για να δομηθεί μια άλλη μεταβλητή (`answer`), η οποία επίσης περιέχει μια ακολουθία χαρακτήρων. Η μεταβλητή `answer` δομείται, συρράπτοντας (concatenation) κείμενο 'Hello ' με το κείμενο της `name`. Έπειτα προστίθεται και το κείμενο που αποτελείται από έναν μόνο χαρακτήρα, τον '!'. Η παραπάνω λειτουργία είναι όμοια με αυτήν της παράθεσης για τους πίνακες που συναντήσαμε σε προηγούμενη ενότητα, μόνο που εδώ αφορά σε ακολουθίες (σειρές) χαρακτήρων. Στο τέλος εμφανίζεται το περιεχόμενο της μεταβλητής `answer`, η οποία περιλαμβάνει και το όνομα που δώσαμε.

7.2 Συνάρτηση

Είδαμε ότι στο MATLAB υπάρχουν κάποιες ενσωματωμένες συναρτήσεις. Μας δίνεται η δυνατότητα να προγραμματίσουμε και καινούριες συναρτήσεις που εμείς θέλουμε. Τα αρχεία συναρτήσεων (function files) είναι επίσης m-αρχεία, αλλά η πρώτη λέξη που περιέχουν πρέπει να είναι `function`. Όπως και σε άλλες γλώσσες προγραμματισμού, τα αρχεία συναρτήσεων λαμβάνουν εξωτερικά ορίσματα, τα οποία περιέχονται σε παρενθέσεις αμέσως μετά το όνομα της συνάρτησης, και μπορούν να έχουν επιστρεφόμενη τιμή. Για παράδειγμα, για να προγραμματίσουμε μια συνάρτηση που υπολογίζει το λογάριθμο ενός πραγματικού αριθμού με βάση έναν οποιοδήποτε άλλο θετικό πραγματικό αριθμό, μπορούμε να γράψουμε τις ακόλουθες γραμμές σε ένα αρχείο με όνομα 'logN.m':

```
function y=logN(base,argument)

%This function calculates the logarithm
%of an array of real numbers (argument)
%using a positive real number as the base.
%Input: base and argument

if (base>0)
   y=log10(argument)./log10(base);
else
   disp(['The base ' num2str(base) ' is not positive.']);
```

```
end
```

Μπορούμε να καλέσουμε τη συνάρτηση, για να υπολογίσουμε το λογάριθμο μιας σειράς δυνάμεων του 10 με βάση το 2, γράφοντας στη γραμμή εντολών:

```
» logN(2,[10 100 1000])

ans =

    3.3219    6.6439    9.9658
```

Το αποτέλεσμα επαληθεύεται με την κλήση της ενσωματωμένης συνάρτησης του MATLAB `log2`, η οποία εκτελεί την ίδια λειτουργία:

```
» log2([10 100 1000])

ans =

    3.3219    6.6439    9.9658
```

Στην παραπάνω περίπτωση η συνάρτηση δέχεται ως είσοδο δύο ορίσματα, από τα οποία το ένα μπορεί να αποτελεί πίνακα αριθμών. Η έξοδος της συνάρτησης είναι μια πραγματική τιμή που αποθηκεύεται στη μεταβλητή `y` (επιστρεφόμενη τιμή). Παρατηρούμε ότι και η επιστρεφόμενη τιμή έχει τις διαστάσεις του ορίσματος. Στον παραπάνω υπολογισμό του λογαρίθμου με βάση το 2 ενός μονοδιάστατου πίνακα με τρεις δυνάμεις του 10, το αποτέλεσμα ήταν επίσης ένας μονοδιάστατος πίνακας με τρεις τιμές. Αυτή η δυνατότητα υπάρχει, γιατί χρησιμοποιήσαμε την τελεία πριν από τον τελεστή της διαίρεσης.

Η συνάρτηση θα μπορούσε να μην έχει καμιά ή περισσότερες από μια επιστεφόμενες τιμές. Για παράδειγμα, η παρακάτω συνάρτηση επιστρέφει την επιφάνεια και τον όγκο μιας σφαίρας, λαμβάνοντας ως είσοδο την ακτίνα της σφαίρας. Γράφουμε τις ακόλουθες εντολές σε ένα αρχείο με το όνομα ‘sphere_calc.m’:

```
function [s, v]=sphere_calc(r)

%This function calculates the surface
%and the volume of different spheres.
%Input: radius of spheres

s=4*pi*r.^2;
v=4/3*pi*r.^3;
```

Μπορούμε να υπολογίσουμε με τη συνάρτηση αυτή την επιφάνεια και τον όγκο είτε για μια ακτίνα:

```
» [s,v]=sphere_calc(4)

s =

  201.0619


v =

  268.0826
```

είτε, αν ως όρισμα χρησιμοποιήσουμε έναν μονοδιάστατο πίνακα, για όλες τις τιμές του πίνακα:

```
» [s,v]=sphere_calc([2 4])

s =

   50.2655  201.0619


v =

   33.5103  268.0826
```

Προσοχή πρέπει να δώσουμε στην περίπτωση που καλέσουμε τη συνάρτηση από τη γραμμή εντολών, χωρίς να έχουμε καθορίσει τις μεταβλητές, στις οποίες θα σωθούν οι επιστρεφόμενες τιμές:

```
» sphere_calc([2 4])

ans =

   50.2655  201.0619
```

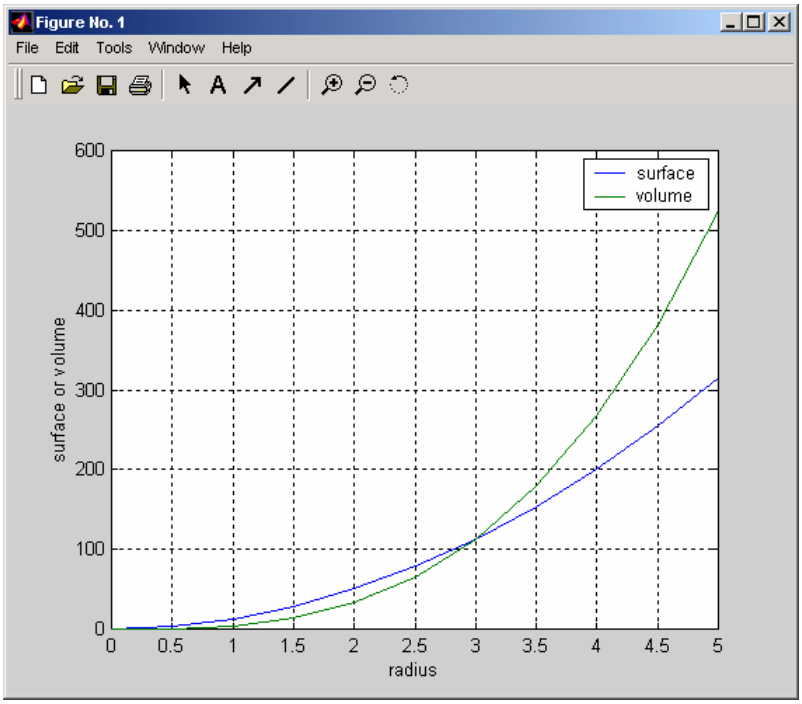
Είναι φανερό ότι τώρα επιστρέφεται μόνο η πρώτη από τις επιστρεφόμενες μεταβλητές, στη μεταβλητή `ans`.

Χρησιμοποιώντας τη συνάρτηση ‘sphere_calc’ μπορούμε να σχεδιάσουμε (Σχήμα 7) τη μεταβολή της επιφάνειας και του όγκου μιας σφαίρας με την ακτίνα της:

```
» r=0:0.5:5;
» [s,v]=sphere_calc(r);
» plot(r,s,r,v)
» legend('surface','volume')
```

```
» xlabel('radius')
» ylabel('surface or volume')
» grid
```

Με την εντολή `legend` δημιουργούμε ένα υπόμνημα για το γράφημα, δηλαδή δίνουμε στις γραμμές που σχεδιάστηκαν έναν τίτλο, παραθέτοντας τόσες ακολουθίες χαρακτήρων μέσα σε απλά εισαγωγικά, όσες και οι σχεδιαζόμενες γραμμές (δυο στο συγκεκριμένο παράδειγμα).



Σχήμα 7. Μεταβολή του όγκου και της επιφάνειας μιας σφαίρας με την ακτίνα της.

Ως τελευταία παρατήρηση για τις συναρτήσεις πρέπει να σημειωθεί ότι μπορούμε να καλέσουμε μια συνάρτηση μέσα από οποιαδήποτε άλλη συνάρτηση (το έχουμε ήδη κάνει καλώντας ενσωματωμένες συναρτήσεις του MATLAB) αλλά και μέσα από τον εαυτό της (επαναληπτική κλήση, recursive call).

7.3 Δομές προγραμματισμού

Οι βασικές δομές προγραμματισμού είναι οι δομές ελέγχου και οι δομές επανάληψης (βρόχοι).

7.3.1 Δομές διακλάδωσης με συνθήκη (conditional branching)

Στη συνάρτηση 'logN' χρησιμοποιήθηκε μια δομή ελέγχου υπό συνθήκη που αρχίζει με την εντολή `if`, χωρίς να αναλυθεί. Στην ενότητα αυτή περιλαμβάνονται περισσότερα στοιχεία για τη σχετική δομή.

Οι δομές ελέγχου με συνθήκη χρησιμοποιούνται στην περίπτωση που πρέπει να ληφθεί μια απόφαση κατά την εκτέλεση του προγράμματος. Ανάλογα με την απόφαση που λαμβάνεται εκτελείται διαφορετικό τμήμα του προγράμματος, δηλαδή αλλάζει η ροή του. Η βασική σύνταξη μιας δομής ελέγχου είναι:

```
if condition (λογική έκφραση)
      statements 1 (εντολές)
else
      statements 2 (εντολές)
end
```

Αν η λογική έκφραση `condition` είναι αληθής, τότε εκτελείται το πρώτο σύνολο εντολών (`statements 1`), ενώ σε αντίθετη περίπτωση εκτελείται το δεύτερο σύνολο εντολών (`statements 2`):

```
if x>5
   disp(['x larger than 5'])
else
   disp(['x smaller than or equal to 5'])
end
```

Σε ορισμένες περιπτώσεις παραλείπεται το δεύτερο τμήμα της δομής, οπότε στην περίπτωση που η λογική έκφραση γίνεται ψευδής δεν εκτελείται κάποια εντολή αλλά συνεχίζεται η ροή του προγράμματος. Στο παρακάτω παράδειγμα, αν η μεταβλητή `x` είναι αρνητικός αριθμός παίρνουμε απλώς μια προειδοποίηση:

```
if x<0
   disp(['Warning: x is negative, the root will be complex'])
end
a=sqrt(x);
```

Επίσης, υπάρχουν περιπτώσεις όπου έχουμε περισσότερες διακλαδώσεις (τμήματα) της δομής ελέγχου, γιατί θέλουμε να ελέγξουμε περισσότερες λογικές συνθήκες. Οι παρακάτω γραμμές κατατάσσουν την τιμή τάσης που καταχωρήθηκε στη μεταβλητή `V` σε τρεις περιοχές, 'υψηλή' (μεγαλύτερη από 100 μονάδες μέτρησης), 'μεσαία' (μεταξύ 50 και 100 μονάδων μέτρησης) και 'χαμηλή' (μικρότερη από 50 μονάδες μέτρησης):

```
if V>=100
   disp('Danger: high voltage!');
elseif (V>=50)&(V<100)
   disp('Warning: medium voltage.');
else
   disp('Low voltage, safe to proceed.');
end
```

Όπως φαίνεται στο τελευταίο παράδειγμα, σε μια λογική έκφραση εκτός από τελεστές σύγκρισης (Πίνακας 2) χρησιμοποιούμε και τους λογικούς τελεστές (Πίνακας 3). Η τιμή 0 σημαίνει 'ψευδής' για το MATLAB. Οποιοσδήποτε άλλος πραγματικός αριθμός σε μια λογική έκφραση έχει τη λογική τιμή 'αληθής'.

Πίνακας 2: Τελεστές σύγκρισης

Πράξη	Τελεστής	Παράδειγμα λογικής έκφρασης (αποτέλεσμα)
έλεγχος ισότητας	`==`	`3 == 2` (0, δηλ. ψευδής)
έλεγχος ανισότητας	`~=`	`3 ~= 2` (1, δηλ. αληθής)
μικρότερο	`<`	`3 < 2` (0, δηλ. ψευδής)
μεγαλύτερο	`>`	`3 > 2` (1, δηλ. αληθής)
μικρότερο ή ίσο	`<=`	`3 <= 2` (0, δηλ. ψευδής)
μεγαλύτερο ή ίσο	`>=`	`3 >= 2` (1, δηλ. αληθής)

Πίνακας 3: Λογικοί τελεστές

Πράξη	Τελεστής	Παράδειγμα λογικής έκφρασης (αποτέλεσμα)
λογική σύζευξη	`&`	`(3 == 2) & (4 == 4)` (0, δηλ. ψευδής)
λογική διάζευξη	`|`	`(3 == 2) | (4 == 4)` (1, δηλ. αληθής)
αποκλειστική διάζευξη	`XOR(a,b)`	`XOR(1,0)` (1, δηλ. αληθής)
λογική άρνηση	`~`	`~(3 >= 2)` (0, δηλ. ψευδής)

7.3.2 Δομές επανάληψης με συνθήκη (conditional loops)

Η εκτέλεση μιας ή περισσοτέρων εντολών (και ενσωματωμένων ή δικών μας συναρτήσεων) μπορεί να γίνει στο MATLAB με τη χρησιμοποίηση των δύο δομών επανάληψης με συνθήκη, του βρόχου `for` και του βρόχου `while`.

7.3.2.1 Βρόχος `for`

Αυτός ο βρόχος επανάληψης χρησιμοποιείται, όταν απαιτούνται επαναλήψεις συγκεκριμένου πλήθους. Δεν είναι απαραίτητο για τον προγραμματιστή να γνωρίζει εκ των προτέρων το πλήθος των επαναλήψεων. Μπορεί αυτό να αλλάζει κατά την εκτέλεση του προγράμματος. Στο παρακάτω παράδειγμα, το οποίο σώζουμε στο αρχείο κειμένου (script file), 'for_loop.m', δίνεται ένα πρόγραμμα με το οποίο μπορούμε να υπολογίσουμε (με τη συνάρτηση

'sphere_calc.m' που γράψαμε παραπάνω) την επιφάνεια και τον όγκο όσων σφαιρών εμείς επιλέξουμε. Αρχικά, δηλαδή, το πρόγραμμα ζητάει από εμάς το πλήθος των επαναλήψεων κλήσης της συνάρτησης:

```
% The program is an example of the for-loop.
% Calls repeatedly the function 'sphere_calc.m'
nspheres = input('How many spheres do you want to calculate? ');
for i=1:nspheres
   r = input(['Please enter the radius of sphere ' ...
         num2str(i) ' : ']);
   [s,v]=sphere_calc(r);
   answer_surf=['The surface of sphere ' ...
         num2str(i) ' is ' num2str(s) ' square units.'];
   disp(answer_surf)
   answer_volu=['The volume of sphere ' ...
         num2str(i) ' is ' num2str(v) ' cubic units.'];
   disp(answer_volu)
end
```

Πριν πούμε περισσότερα για το βρόχο `for` αξίζει να τονιστούν δυο σημεία του παραπάνω παραδείγματος. Το πρώτο είναι ότι, επειδή στο MATLAB, κάθε εντολή πρέπει να γράφεται σε μια γραμμή, μπορούμε να χρησιμοποιήσουμε τα αποσιωπητικά `...`, για να δηλώσουμε ότι η εντολή συνεχίζεται και στην επόμενη γραμμή. Επίσης, στο παραπάνω παράδειγμα γίνεται χρήση της εντολής `num2str`, η οποία επιτρέπει τη μετατροπή ενός αριθμού σε ακολουθία χαρακτήρων (που σχηματίζεται με τα ψηφία που αποτελούν τον αριθμό).

Ας δούμε, επομένως, ποια είναι η γενική σύνταξη ενός βρόχου `for`:

```
for variable = expression (μεταβλητή = έκφραση)
      statements (εντολές)
end
```

Η τιμή της μεταβλητής `variable` που δίνεται στην έκφραση `expression` μπορεί να είναι είτε ένα διάνυσμα είτε ένας πίνακας. Κάθε τιμή της έκφρασης σώζεται στη μεταβλητή και κατόπιν εκτελούνται οι εντολές `statements`. Αυτή η επανάληψη της εκτέλεσης των εντολών μεταξύ του `for` και του `end` συνεχίζεται ώσπου η μεταβλητή να εξαντλήσει τις τιμές της έκφρασης. Παρακάτω δίνονται μερικά παραδείγματα εκφράσεων:

```
for i=1:10
   statements
end

for i=20:-1:10,
   statements
end

for i=[3 85 400]
   statements
```

```
end

for i=[1 8 12; 3 15 34]
   statements
end
```

Στην τελευταία περίπτωση, όπου η μεταβλητή παίρνει τιμές από έναν δισδιάστατο πίνακα, σε κάθε επανάληψη παίρνει τις τιμές μιας στήλης του πίνακα. Άρα, ο τελευταίος βρόχος θα εκτελεστεί τρεις φορές.

Σε αυτό το σημείο πρέπει να τονιστεί ότι σε όλα τα παραπάνω παραδείγματα η φανταστική μονάδα παύει να συμβολίζεται με το `i`, το οποίο αποτελεί πλέον μεταβλητή που μπορεί να πάρει οποιαδήποτε τιμή (πολλές φορές ονομάζεται και μετρητής, από τον αγγλικό όρο counter).

7.3.2.2 Βρόχος `while`

Στην περίπτωση που δεν απαιτείται η εκτέλεση κάποιων εντολών για συγκεκριμένο πλήθος επαναλήψεων, αλλά θέλουμε αυτές να εκτελούνται για όσο διάστημα ισχύουν συγκεκριμένες προϋποθέσεις (π.χ. μια λογική έκφραση να είναι αληθής ή ψευδής), χρησιμοποιούμε το βρόχο `while`:

```
while condition (λογική έκφραση)
      statements (εντολές)
end
```

Οι εντολές εκτελούνται συνεχώς, όσο η λογική έκφραση `condition` είναι αληθής. Είναι φανερό ότι δεν μπορούμε σε όλες τις περιπτώσεις χρήσης του βρόχου αυτού να προβλέψουμε το πλήθος των επαναλήψεων κάθε φορά που τρέχουμε το πρόγραμμα. Αυτές μπορεί να είναι από καμια, αν η λογική έκφραση δεν είναι ποτέ αληθής, π.χ.

```
while 1==2
   statements
end
```

έως άπειρες σε αριθμό, αν η λογική έκφραση είναι διαρκώς αληθής, π.χ.

```
while 1==1
   statements
end
```

Η τελευταία περίπτωση ονομάζεται “ατέρμονος βρόχος“ και πρέπει να αποφεύγεται. Παρακάτω δίνεται ένα παράδειγμα χρήσης του βρόχου `while`. Πρόκειται για μια συνάρτηση,

η οποία υπολογίζει πόσοι όροι απαιτούνται, ώστε το άθροισμα $\sum_{n=1}^{m}\frac{1}{n}$ να είναι μικρότερο από έναν αριθμό που δίνεται ως όρισμα της συνάρτησης:

```
function count=while_loop(limit)

% The function is an example of the while-loop.
% It calculates the number of terms in the \sum(1/n),
% so that it remains smaller than a specific number.
% Input: upper limit of the sum

sum=0;
count=0;

while sum<limit
   count=count+1;
   sum=sum+1/count;
end

count=count-1;
```

Για παράδειγμα, το άθροισμα παραμένει μικρότερο του 2, αν χρησιμοποιηθούν μέχρι και οι 3 πρώτοι όροι, ενώ είναι μικρότερο του 6 για 226 όρους!

7.3.2.3 Φωλιασμένοι βρόχοι (nested loops)

Τις δομές επανάληψης μπορούμε να τις χρησιμοποήσουμε τη μια μέσα στην άλλη, σχηματίζοντας φωλιασμένους (ή πολλαπλούς) βρόχους. Ένα παράδειγμα είναι ο σχηματισμός ενός άνω τριγωνικού πίνακα διαστάσεων N×N. Αν αποθηκεύσουμε τον παρακάτω κώδικα στο αρχείο 'nested_loops.m' μπορούμε να ελέγξουμε τη λειτουργία του.

```
% This program creates an upper
% triangular matrix

N=input('Please enter size of matrix : ');
pinakas=zeros(N);
for i=1:N
   for j=1:N
      if (i<j)
         pinakas(i,j)=1;
      end
   end
end
pinakas
```

```
» nested_loop
Please enter size of matrix : 4

pinakas =
```

```
     0     1     1     1
     0     0     1     1
     0     0     0     1
     0     0     0     0
```

7.3.2.4 Εντολή `break`

Η εκτέλεση ενός βρόχου μπορεί να διακοπεί από το εσωτερικό του, καλώντας την εντολή `break`. Με το παρακάτω παράδειγμα βρίσκουμε το μικρότερο πρώτο αριθμό σε ένα διάστημα ακεραίων τιμών. Η εκτέλεση του βρόχου διακόπτεται, όταν βρεθεί αυτός ο αριθμός. Για τον έλεγχο των αριθμών χρησιμοποιούμε την ενσωματωμένη συνάρτηση `isprime` του MATLAB, η οποία επιστρέφει τιμή 1 (‘αληθής’) στην περίπτωση που το όρισμα κλήσης της είναι πρώτος αριθμός και τιμή 0 (‘ψευδής’) σε αντίθετη περίπτωση.

```
function p=smallest_prime(range)

% This function finds the smallest prime
% number in a range of natural numbers.
% Input: range of numbers

for p=range(1):range(2)
   if isprime(p)
      break;
   end
end
```

8 Αριθμητική ανάλυση

8.1 Αριθμητική ολοκλήρωση

Η αριθμητική ολοκλήρωση συναρτήσεων μιας μεταβλητής στο MATLAB είναι σχετικά απλή. με τη μέθοδο του Euler. Η προσέγγιση του ολοκληρώματος βασίζεται ουσιαστικά στο άθροισμα των τιμών της συνάρτησης για διάφορες τιμές της μεταβλητής που ισαπέχουν μεταξύ τους. Το τελικό άθροισμα πολλαπλασιάζεται με την απόσταση μεταξύ των τιμών της μεταβλητής (βήμα ολοκλήρωσης), για να προκύψει η αριθμητική τιμή. Όπως είναι φανερό και από τον ορισμό του ολοκληρώματος, όσο πιο μικρό είναι το βήμα που χρησιμοποιούμε μεταξύ των τιμών της μεταβλητής, τόσο πιο κοντά θα είναι η αριθμητική τιμή στην πραγματική τιμή του ολοκληρώματος. Αυτή τη διαπίστωση μπορούμε να την κάνουμε με το παρακάτω παράδειγμα, στο οποίο σχεδιάζουμε την αριθμητική τιμή του ολοκληρώματος $\int_0^{\pi/2} \sin(x)dx$ ως συνάρτηση του βήματος ολοκλήρωσης.

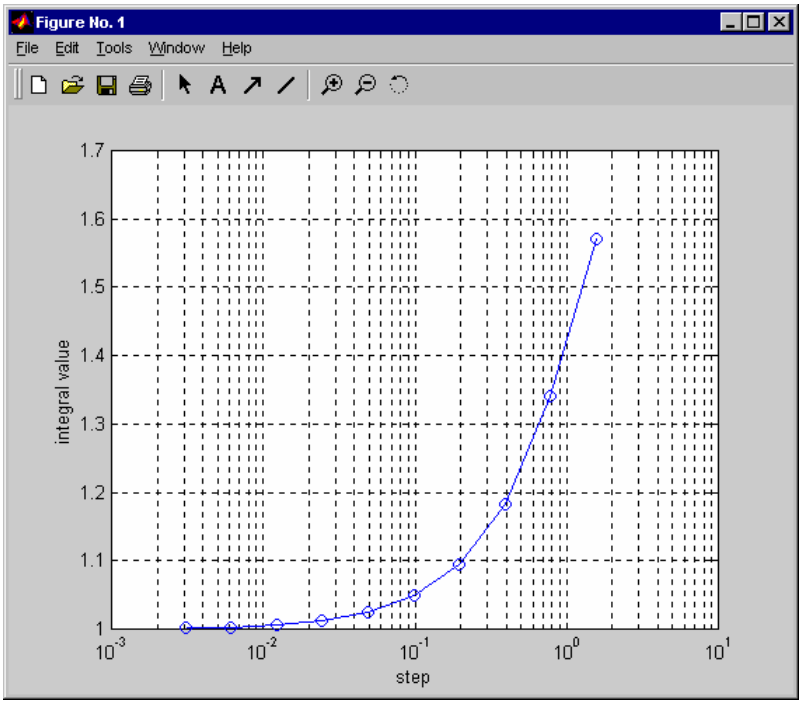
```
% This file examines the Euler method of
% numerical integration by the example of
% the definite integral of sin(x) from 0 to pi/2.

ni=input('Please give the number of iterations: ');
s=zeros(ni,1); %stores the step size
v=zeros(ni,1); %stores the value of integral for each step

step=pi/2; %start with the largest step
for i=1:ni
   x=0:step:pi/2;
   v(i)=sum(sin(x))*step;
   s(i)=step;
   step=step/2;
end

semilogx(s,v,'o-')
grid
xlabel('step')
ylabel('integral value')
```

Είναι γνωστό ότι η τιμή του ολοκληρώματος είναι το 1. Από το Σχήμα 8 φαίνεται ότι μετά από δέκα υποδιπλασιασμούς του βήματος ολοκλήρωσης, η αριθμητική τιμή του ολοκληρώματος είναι πολύ κοντά στην πραγματική τιμή του.



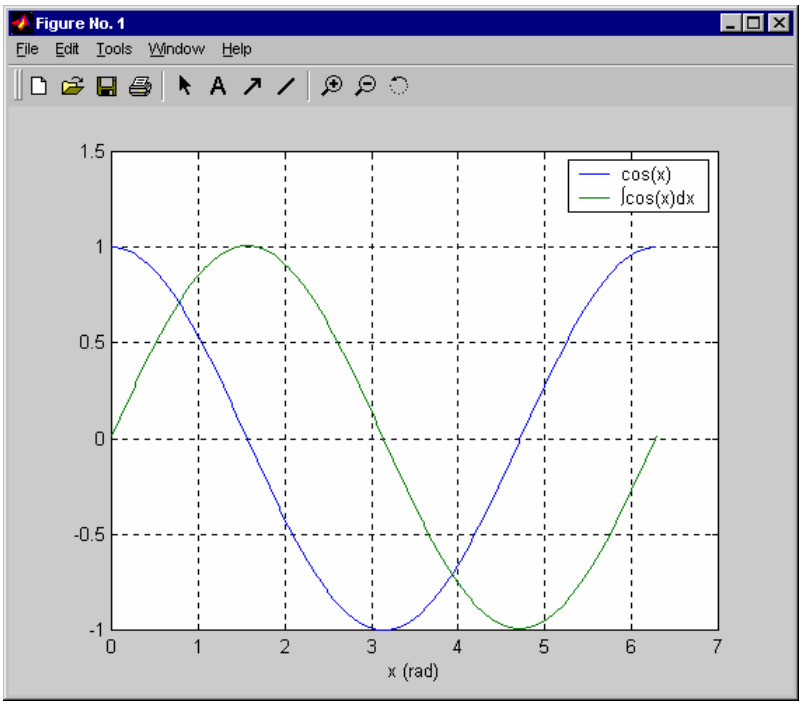
Σχήμα 8. Σύγκλιση της αριθμητικής τιμής ολοκληρώματος προς την πραγματική τιμή.

Ο υπολογισμός του αορίστου ολοκληρώματος $\int_0^y \cos(x)dx$ γίνεται με παρόμοιο τρόπο με τη μέθοδο Euler, χρησιμοποιώντας τη συνάρτηση `cumsum` του MATLAB η οποία υπολογίζει το σωρευτικό άθροισμα ενός διανύσματος γραμμής ή στήλης:

```
» step=pi/200;
» x=0:step:2*pi;
» y=cumsum(cos(x))*step;
» plot(x,cos(x),x,y)
» grid
» xlabel('x (rad)')
» legend('cos(x)','\intcos(x)dx')
```

Όπως φαίνεται και στο Σχήμα 9, το αριθμητικό ολοκλήρωμα του συνημιτόνου πλησιάζει πολύ προς το ημίτονο που είναι το πραγματικό ολοκλήρωμα.

Το MATLAB έχει και ενσωματωμένες συναρτήσεις για αριθμητική ολοκλήρωση με τη μέθοδο του τραπεζίου, τις `trapz` και `cumtrapz`, οι οποίες λειτουργούν με τον τρόπο που λειτουργούν οι συναρτήσεις `cum` και `cumsum`, αντίστοιχα, μόνο που κάνουν και τις απαραίτητες διορθώσεις στα άκρα του αθροίσματος, ώστε να ισχύει ο κανόνας του τραπεζίου.



Σχήμα 9. Η συνάρτηση $\cos(x)$ και το αριθμητικό της ολοκλήρωμα.

8.2 Αριθμητική διαφόριση

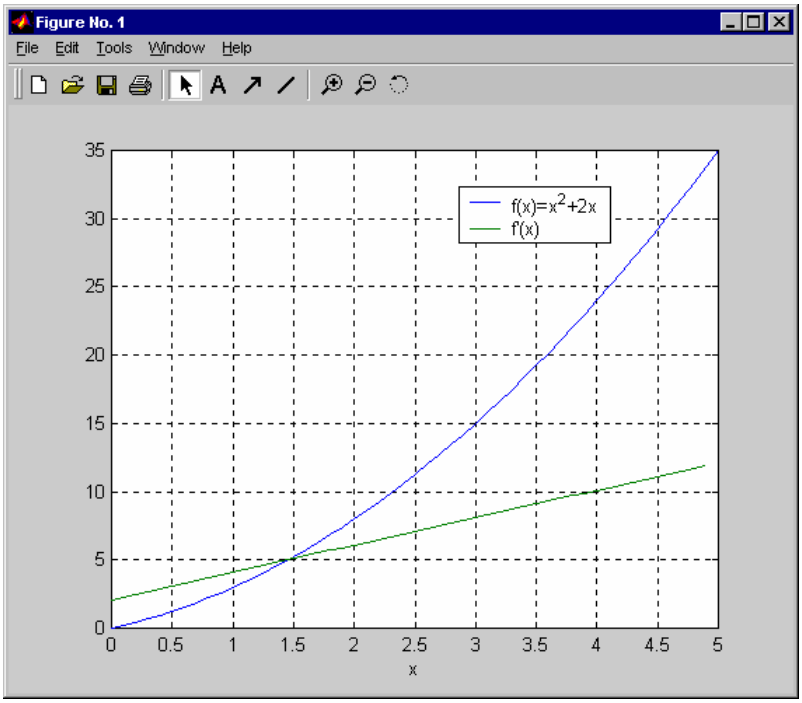
Η αριθμητική διαφόριση στο MATLAB γίνεται με την εντολή diff, η οποία σε ένα διάνυσμα γραμμής ή στήλης βρίσκει τη διαφορά μεταξύ διαδοχικών στοιχείων του. Έτσι, μπορούμε να πούμε ότι, αν η απόσταση μεταξύ δύο στοιχείων του διανύσματος είναι πολύ μικρή, η αριθμητική προσέγγιση της παραγώγου μπορεί να γίνει με τη μέθοδο της εμπρόσθιας διαφοράς:

$$\left.\frac{df}{dx}\right|_{x=x_0} \simeq \frac{f(x_0+h)-f(x_0)}{h}$$

Ως παράδειγμα, θα χρησιμοποιήσουμε τη συνάρτηση x^2+2x της οποίας γνωρίζουμε ότι η παράγωγος είναι η συνάρτηση $2x+2$. Επειδή θα χρησιμοποιήσουμε ένα διάνυσμα τιμών της συνάρτησης και θα πάρουμε διαφορές μεταξύ των διαδοχικών στοιχείων, το διάνυσμα της παραγώγου που θα προκύψει, θα έχει ένα στοιχείο λιγότερο, και γι΄αυτό πρέπει να δοθεί ιδιαίτερη προσοχή στη σχεδίαση του αποτελέσματος:

```
» x=0:step:5;
» y=x.^2+2*x;
» dydx=diff(y)/step;
» plot(x,y,x(1:end-1),dydx)
» grid
» xlabel('x')
» legend('f(x)=x^2+2x','f''(x)')
```

Μπορούμε εύκολα να επαληθεύσουμε από το Σχήμα 10 ότι η διαφόριση της συνάρτησης δίνει το σωστό αποτέλεσμα.



Σχήμα 10. Πολυωνυμική συνάρτηση και η αριθμητική προσέγγιση της παραγώγου της.

9 Διαχείριση αρχείων δεδομένων

9.1 Είδη αρχείων

Το MATLAB μπορεί να αποθηκεύσει σε και να ανακτήσει δεδομένα από αρχεία δυο τύπων, τα δυαδικά (binary) αρχεία και τα αρχεία κειμένου (ASCII).

Τα δυαδικά αρχεία αποθηκεύουν αριθμούς σε δυαδική μορφή, δηλαδή ως σειρές των ψηφίων 1 και 0 (bits). Ανάλογα με το μέγεθος του αριθμού που θα αποθηκευτεί χρησιμοποιείται και το αντίστοιχο πλήθος από bits που σχηματίζουν bytes ανά ομάδες των οκτώ. Έτσι, ένας αριθμός μικρότερος από το 256 μπορεί να αποθηκευτεί με ένα byte. Γενικά, με n bits η μεγαλύτερη δεκαδική τιμή που μπορούμε να παραστήσουμε είναι $2^n - 1$. Στην πραγματικότητα όλες οι μεταβλητές στο MATLAB παριστάνονται ως αριθμοί κινητής υποδιαστολής (floating point numbers), χρησιμοποιώντας 8 bytes ο καθένας, και οι τιμές που παίρνουν είναι πολύ μεγαλύτερες από $2^{32} - 1$.

Αντίθετα, στα αρχεία ASCII κάθε αριθμός αποθηκεύεται ως ακολουθία των ψηφίων που τον αποτελούν. Αυτό γίνεται με τη βοήθεια του πίνακα ASCII, που είναι μια μονοσήμαντη απεικόνιση των χαρακτήρων του πληκτρολογίου ενός Η/Υ (αλλά και κάποιων άλλων εκτός από αυτούς) σε αριθμούς που αποθηκεύονται σε ένα byte. Επομένως, για να αποθηκεύσουμε τον αριθμό 123 σε ένα αρχείο, απαιτούνται 3 byte.

9.2 Αποθήκευση δεδομένων

Η αποθήκευση όλων των μεταβλητών, οι οποίες έχουν κάποια τιμή τη στιγμή που εργαζόμαστε (μπορούμε να τις δούμε όλες με την εντολή `whos`), μπορεί να γίνει με την εντολή

```
save filename
```

Με τον τρόπο αυτόν δημιουργείται ένα δυαδικό αρχείο με το όνομα 'filename.mat', το οποίο μπορεί να διαχειριστεί το MATLAB. Σε περίπτωση που θέλουμε να σώσουμε σε αρχείο μόνο μια μεταβλητή με το όνομα `X` γράφουμε

```
save filename X
```

Το MATLAB δίνει τη δυνατότητα αποθήκευσης μεταβλητών σε αρχεία ASCII, ως εξής

```
save filename X -ASCII
```

Η εντολή αυτή σώζει τα δεδομένα στο αρχείο στη μορφή που είχε η κάθε μεταβλητή, δηλαδή σε γραμμές, στήλες ή πίνακες, αφήνοντας κενό μεταξύ των στηλών. Για παράδειγμα ο πίνακας με τις γωνίες από 0° ως 90° και τα ημίτονά τους σώζεται ως

```
  0.0000000e+000  0.0000000e+000
  1.0000000e+000  1.7452406e-002
  2.0000000e+000  3.4899497e-002
  3.0000000e+000  5.2335956e-002
  4.0000000e+000  6.9756474e-002
  5.0000000e+000  8.7155743e-002
  6.0000000e+000  1.0452846e-001
  7.0000000e+000  1.2186934e-001
  8.0000000e+000  1.3917310e-001
  9.0000000e+000  1.5643447e-001
  . . . . . .  . . . . . .

  8.6000000e+001  9.9756405e-001
  8.7000000e+001  9.9862953e-001
  8.8000000e+001  9.9939083e-001
  8.9000000e+001  9.9984770e-001
  9.0000000e+001  1.0000000e+000
```

Αν γράψουμε απλώς την εντολή `save`, χωρίς όνομα αρχείου, τότε όλες οι μεταβλητές που έχουμε χρησιμοποιήσεια αποθηκεύονται στο δυαδικό αρχείο 'MATLAB.mat'.

9.3 Εισαγωγή δεδομένων

Η εισαγωγή δεδομένων από ένα αρχείο ASCII μπορεί να γίνει πολύ εύκολα στο MATLAB, χρησιμοποιώντας την εντολή load. Αυτό το γεγονός αποτελεί ένα μεγάλο πλεονέκτημα του

MATLAB, γιατί τα περισσότερα πακέτα λογισμικού, ιδιαίτερα τα λογιστικά φύλλα (speadsheets) έχουν τη δυνατότηα εξαγωγής των δεδομένων σε μορφή ASCII. Αν, για παράδειγμα, ο παραπάνω πίνακας των ημιτόνων είχε σωθεί σε ένα αρχείο με το όνομα 'sinfile.dat' και σε δύο στήλες, τότε η είσοδος των δεδομένων αυτών θα γινόταν, γράφοντας

```
» load sinfile.dat
```

Ως αποτέλεσμα αυτής της εντολής, σχηματίζεται μια μεταβλητή με το όνομα `sinfile`, δηλαδή το όνομα του αρχείου χωρίς το επίθεμά του. Η μεταβλητή είναι ένα πίνακας με 91 γραμμές και 2 στήλες και μπορούμε να τη σχεδιάσουμε (ουσιαστικά να σχεδιάσουμε τη συνάρτηση ημιτόνου στο πρώτο τεταρτημόριο) γράφοντας

```
» plot(sinfile(:,1),sinfile(:,2))
```

Είναι δυνατό να διαβάσουμε και να καταχωρήσουμε τα δεδομένα του παραπάνω αρχείου σε μια μεταβλητή με το όνομα που θέλουμε εμείς, έστω `sindata`, ως εξής

```
» sindata=load('sinfile.dat');
```

Αν χρησιμοποιήσουμε την εντολή `load` με όνομα αρχείου χωρίς επίθεμα, τότε θεωρείται ότι αυτό το αρχείο είναι δυαδικό αρχείο τύπου mat. Δηλαδή, αν γράψουμε

```
» load sinfile
```

τότε το MATLAB ψάχνει το αρχείο 'sinfile.mat' για να φορτώσει τις μεταβλητές που είναι αποθηκευμένες σε αυτό. Αν το αρχείο δε βρεθεί στο τρέχον μονοπάτι εμφανίζεται το αντίστοιχο μήνυμα λάθους:

```
» load sinfile
??? Error using ==> load
sinfile.mat: Can't open file.
```

Όπως και στην περίπτωση της εντολής `save`, αν χρησιμοποιήσουμε την εντολή `load` χωρίς το όνομα κάποιου ιδιαίτερου αρχείου, υπονοείται το αρχείο 'MATLAB.mat'.

Τέλος, πρέπει να σημειωθεί ότι το MATLAB επιτρέπει τη διαχείριση αρχείων με εντολές της γλώσσας προγραμματισμού C, με μικρές μόνο αποκλίσεις στον τρόπο λειτουργίας τους.

10 Τρισδιάστατα γραφικά

Έχουμε ήδη συναντήσει τη χρήση εντολών του MATLAB, όπως η `plot` και η `semilogx` για τη σχεδίαση γραφημάτων στις δύο διαστάσεις. Επιπλέον υπάρχουν οι εντολές `semilogy` και `loglog`, όταν θέλουμε ο κατακόρυφος άξονας ή και οι δυο άξονες του γραφήματος, αντίστοιχα, να είναι σε λογαριθμική κλίμακα. Υπάρχουν και άλλες εντολές που χρησιμοποιούνται στο MATLAB για τα δισδιάστατα γραφικά, τις οποίες μπορούμε να δούμε πληκτρολογώντας `help graph2d` στη γραμμή εντολών. Όμως η λειτουργία του MATLAB που έπαιξε σημαντικό ρόλο στη γρήγορη διάδοσή του είναι η εύκολη δημιουργία τρισδιάστατων γραφημάτων που βοηθούν στην κατανόηση πολλών προβλημάτων και στην οπτικοποίηση-απεικόνιση (visualization) δεδομένων.

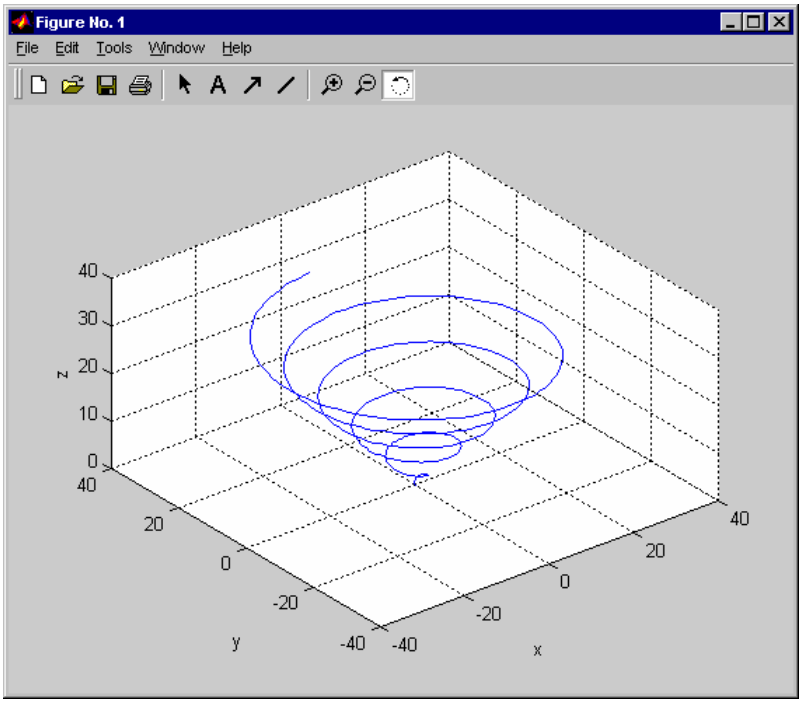
Η εντολή `plot3` χρησιμοποιείται για το σχεδιασμό καμπυλών στις τρεις διαστάσεις. Για παράδειγμα, με τις παρακάτω εντολές μπορούμε να σχεδιάσουμε (Σχήμα 11) μια έλικα μεταβλητής ακτίνας:

```
» t=0:pi/50:10*pi;
» plot3(t.*sin(t),t.*cos(t),t);
» grid;xlabel('x');ylabel('y');zlabel('z')
```

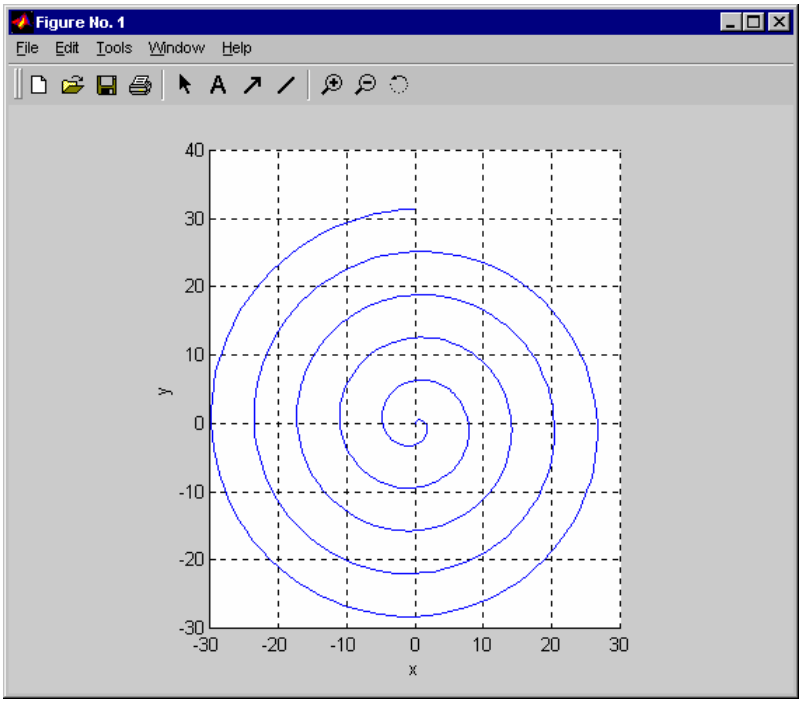
Στα τρισδιάστατα γραφικά μπορούμε να αλλάξουμε τη γωνία προβολής με την εντολή `view`, η οποία δέχεται ένα όρισμα. Το όρισμα μπορεί να είναι διάνυσμα γραμμής δύο στοιχείων, από τα οποία το πρώτο αντιστοιχεί στη γωνία του αζιμουθίου και το δεύτερο στη γωνία ανύψωσης της θέσης του παρατηρητή. Οι δύο γωνίες δίνονται σε μοίρες. Για παράδειγμα, αν γράψουμε

```
» view([0 90])
```

τότε βλέπουμε την προβολή του σχήματος στο επίπεδο xy (Σχήμα 12), ενώ με την εντολή `view([90 0])` παίρνουμε την προβολή στο επίπεδο yz και με την εντολή `view([180 0])` στο επίπεδο xz. Το αποτέλεσμα του Σχήματος 12 μπορούμε να το πάρουμε και γράφοντας `view(2)`.



Σχήμα 11. Σχεδίαση καμπυλών στις τρεις διαστάσεις.



Σχήμα 12. Προβολή του σχήματος στο επίπεδο xy.

Αλλά και η απεικόνιση συναρτήσεων δυο μεταβλητών είναι σχετικά εύκολη με το MATLAB. Για παράδειγμα μπορούμε να σχεδιάσουμε τη συνάρτηση $f(x,y)=x^2+y^2$, η οποία αντιστοιχεί σε ένα ελλειπτικό παραβολοειδές, όπως παρακάτω. Αρχικά ορίζουμε το διάστημα των τιμών των μεταβλητών. Στην περίπτωση αυτή το διάστημα [-1 1] επιλέγεται και για τις

δυο μεταβλητές. Η σχεδίαση γίνεται για βήμα 0.05, δηλαδή επιλέγονται 41 σημεία για καθεμιά μεταβλητή:

```
» x=-1:0.05:1;
» y=x;
```

Έπειτα σχηματίζουμε ένα πλέγμα (κάναβο) τιμών για καθεμιά από τις μεταβλητές με την εντολή `meshgrid`:

```
» [X,Y]=meshgrid(x,y);
```

Οι πίνακες που προκύπτουν έχουν διάσταση 41×41, γιατί το ίδιο πλήθος τιμών έχει επιλεγεί και για τις δυο μεταβλητές. Στη γενική περίπτωση οι πίνακες δεν είναι τετραγωνικοί. Υπολογίζουμε την τιμή της συνάρτησης σε κάθε σημείο του πλέγματος:

```
» Z=X.^2+Y.^2;
```

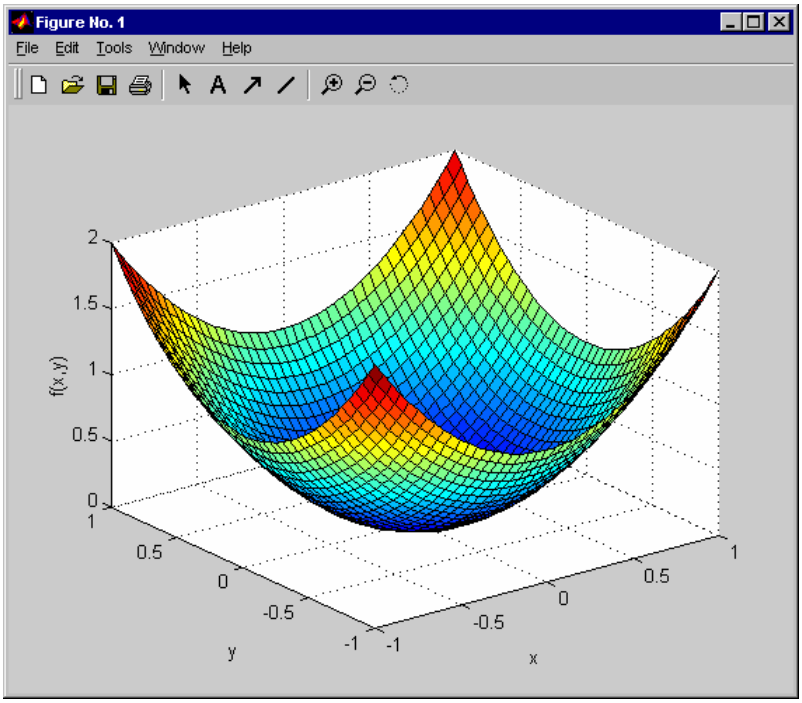
και σχεδιάζουμε με την εντολή `surf` (Σχήμα 13):

```
» surf(X,Y,Z)
» xlabel('x');ylabel('y');zlabel('f(x,y)')
```

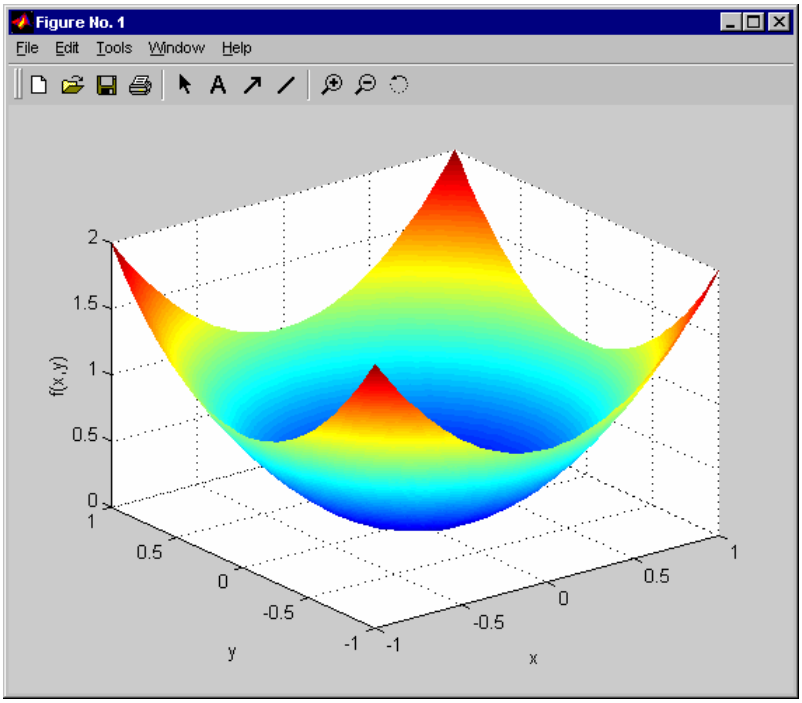
Μια πολύ χρήσιμη εντολή για την απεικόνιση επιφανειών είναι η `shading` η οποία μπορεί να πάρει ως όρισμα τις τιμές `flat`, `interp` και `faceted`. Η τελευταία τιμή είναι η προεπιλεγμένη (default) και αντιστοιχεί στην εικόνα του Σχήματος 13. Αν όμως γράψουμε

```
» shading('interp')
```

τότε παίρνουμε την εικόνα του Σχήματος 14. Η διαφορά της είναι ότι τώρα η μεταβολή των χρωμάτων είναι πιο ομαλή μεταξύ των σημείων του κανάβου.



Σχήμα 13. Τρισδιάστατο γράφημα συνάρτησης.

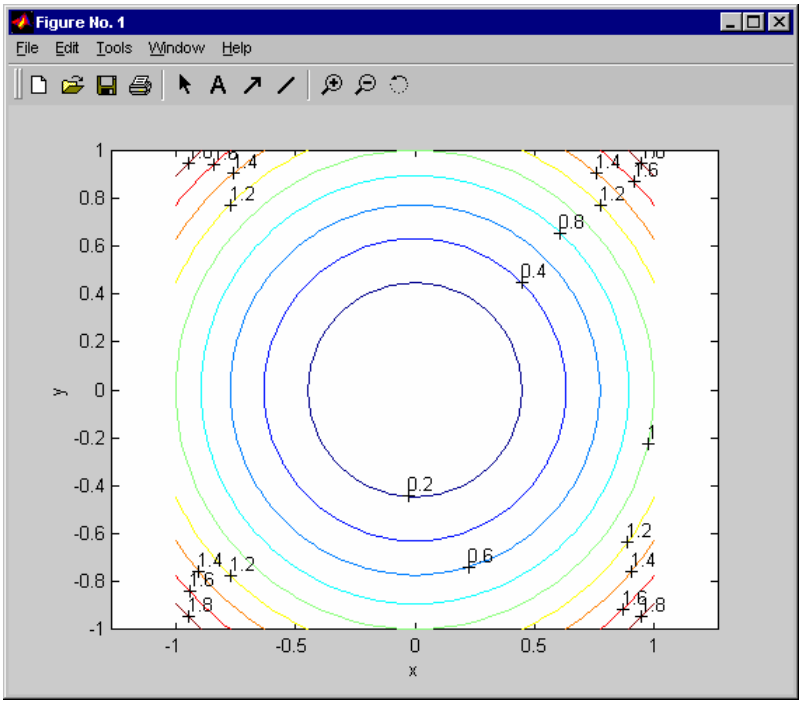


Σχήμα 14. Τρισδιάστατο γράφημα συνάρτησης με την ιδιότητα `shading('interp')`.

Τις ισοϋψείς (σταθμικές) καμπύλες της επιφάνειας που σχηματίζεται μπορούμε να δούμε με την εντολή `contour` (Σχήμα 15). Σχεδιάζουμε αρχικά τις ισοϋψείς και σώζουμε τα στοιχεία τους στη μεταβλητή `c`. Το σχήμα που προκύπτει σχεδιάζεται στο επίπεδο xy σε άξονες με διαφορετικό βήμα, οπότε οι κύκλοι εμφανίζονται ως ελλείψεις. Πρέπει να κάνουμε το βήμα

στους δυο άξονες ίδιο και αυτό επιτυγχάνεται με την εντολή `axis('equal')`. Η επόμενη εντολή δίνει τίτλο στους άξονες x και y αλλά σημειώνει και την τιμή της κάθε καμπύλης:

```
» c=contour(X,Y,Z);
» axis('equal')
» xlabel('x');ylabel('y');clabel(c)
```



Σχήμα 15. Σχεδίαση ισοϋψών (σταθμικών) καμπυλών.